AUTONOMOUS TASK-BASED EVOLUTIONARY DESIGN OF MODULAR ROBOTS

Reem J. Alattas

Under the Supervision of Dr. Tarek M. Sobh

Co-advised by Dr. Sarosh Patel

DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

AND ENGINEERING

THE SCHOOL OF ENGINEERING

UNIVERSITY OF BRIDGEPORT

CONNECTICUT

November, 2018

AUTONOMOUS TASK-BASED EVOLUTIONARY DESIGN

OF MODULAR ROBOTS

Reem J. Alattas

Under the Supervision of Dr. Tarek M. Sobh

Co-advised by Dr. Sarosh Patel

Approvals

Committee Members

Name	Signature	Date
Dr. Tarek M. Sobh	T. Sobh	Nov 29, 2018
Dr. Sarosh Patel	Sarosh	Nov 29, 2018
Dr. Xingguo Xiong	Xingguo Xiong	Nov. 29, 2018
Dr. Miad Faezipour	Miad Faezipour	Nov. 29, 2018
Dr. Kiwon Sohn	Kiwon Sohn	Nov 29, 2018

Ph.D. Program Coordinator

Dr. Khaled M. Elleithy		11/29/18

Chairman, Computer Science and Engineering Department

Dr. Ausif Mahmood	Ausif Mahmood	11-29-2018

Dean, School of Engineering

Dr. Tarek M. Sobh	T. Sobh	Nov 29, 2018

AUTONOMOUS TASK-BASED EVOLUTIONARY DESIGN OF MODULAR ROBOTS

AUTONOMOUS TASK-BASED EVOLUTIONARY DESIGN OF MODULAR ROBOTS

ABSTRACT

In an attempt to solve the problem of finding a set of multiple unique modular robotic designs that can be constructed using a given repertoire of modules to perform a specific task, a novel synthesis framework is introduced based on design optimization concepts and evolutionary algorithms to search for the optimal design.

Designing modular robotic systems faces two main challenges: the lack of basic rules of thumb and design bias introduced by human designers. The space of possible designs cannot be easily grasped by human designers especially for new tasks or tasks that are not fully understood by designers. Therefore, evolutionary computation is employed to design modular robots autonomously.

Evolutionary algorithms can efficiently handle problems with discrete search spaces and solutions of variable sizes as these algorithms offer feasible robustness to local minima in the search space; and they can be parallelized easily to reducing system runtime. Moreover, they do not have to make assumptions about the solution form.

This dissertation proposes a novel autonomous system for task-based modular robotic design based on evolutionary algorithms to search for the optimal design. The introduced system offers a flexible synthesis algorithm that can accommodate to different

task-based design needs and can be applied to different modular shapes to produce homogenous modular robots.

The proposed system uses a new representation for modular robotic assembly configuration based on graph theory and Assembly Incidence Matrix (AIM), in order to enable efficient and extendible task-based design of modular robots that can take input modules of different geometries and Degrees Of Freedom (DOFs).

Robotic simulation is a powerful tool for saving time and money when designing robots as it provides an accurate method of assessing robotic adequacy to accomplish a specific task. Furthermore, it is difficult to predict robotic performance without simulation. Thus, simulation is used in this research to evaluate the robotic designs by measuring the fitness of the evolved robots, while incorporating the environmental features and robotic hardware constraints.

Results are illustrated for a number of benchmark problems. The results presented a significant advance in robotic design automation state of the art.

DEDICATION

To Mom & Dad.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INRTODUCTION

In an attempt to make machines that make machines, evolutionary computation is applied to build robots that can evolve to accomplish a specific task. A number of studies have demonstrated the feasibility of evolutionary algorithms for generating robotic control and morphology. However, a huge challenge faced was how to manufacture these robots. Therefore, divide and conquer strategy was adopted in terms of employing modular robots to simplify robotic evolution and their implementation in real hardware. Still, designing modular robots is a big challenge especially if the task is not fully understood by human designers and the robot has to function in an unstructured environment.

In this dissertation, we propose a novel framework for automating the design process of modular robots using artificial evolution rather than using traditional hand design approach. This framework has the potential of autonomously designing modular robots that can achieve tasks, while requiring the minimum task related knowledge on the designer side.

1.1 Problem Statement

Designing modular robotic system is a huge challenge, because human designers need to fully understand the task a robot has to accomplish and anticipate environmental changes which can be very difficult. Often, the design process is performed in an ad-hoc manner. Human designers start by exploring a number of ideas on paper, then selecting a few for detailed design in simulation. After evaluating these designs, one is selected to be

implemented physically. When testing the robot, unforeseen problems might occur and require changing the design. Design iterations and human mistakes result in significant financial, temporal, and intellectual losses. In addition, human cognition bias might result in deviating from making good design decisions because people tend to think in certain traditional ways [1]. Therefore, performing significant analysis in simulation is crucial before building a physical robot.

The abovementioned problems introduced by human designers made automated design and synthesis tools more appealing for modular robotic design. Thus, evolutionary algorithms are used in this research to automate the design process of modular robots to solve task-based design problems for tasks with little or no human experience in order to produce an optimal solution that can be implemented physically with confidence.

The proposed evolutionary design framework incorporates the task, surrounding environmental conditions, and modular hardware primitives to automate the synthesis of modular robots. The evolutionary approach continuously designs and builds different robots with improved capabilities. The evolved designs are evaluated in simulation to select the best performers to be implemented physically.

Applying evolutionary algorithms improves the parameters and the structure of the robot and modularity simplifies the implementation and search space because of its discrete nature [2]. The end result of this dissertation is a system that takes the available modules and the task to be performed by the resulting robot as input and produces a robotic structure as output.

1.2 Research Motivation

A big motivation of this work is that evolutionary computation can empower modular robots by allowing them to self-assemble, self-reconfigure, self-repair, and self-reproduce. In order to fully exploit these robotic behaviors, a task-based design framework that maps a set of modules into a robotic structure under specific circumstances should be developed.

Self-assembly is the problem of designing a modular robot, such that each module involves edge binding techniques. Hence when the modules mix, they can bind to form the goal robotic design. On the other hand, self-reconfiguration is the process of autonomously changing a robotic structure from one design to another to form a new morphology that can locomote differently or achieve a different task. Self-repair is a special type of self-reconfiguration that allows a robot to fix its own damage. Finally, self-reproduce allows modular robots to reproduce based on the same robotic design.

Thus far, automatic design is a main step in implementing the previous robotic behaviors, whether to allow a robot to self-assemble using an automatically generated design structure or to self-reconfigure to form a new automatically designed shape that better fits the new task or environmental changes.

1.3 Research Contributions

This dissertation is concerned with designing modular robots autonomously using an evolutionary approach that can surpass hand design approach and addressing the challenges of having minimal task related knowledge.

The main contributions of this work to the state of the art in task-based modular robotic design are as follows:

- Introduction of a practical system for designing homogenous modular robots autonomously. The proposed system can adapt to different modular shapes easily by changing a set of input variables to define the new modular shape. For instance, the input modules can be of cubic, spherical, or triangular shapes. Thus, this system can design modular robots that are composed of homogenous modules only. The goal is to produce an extensible system that considers future growth.
- Validation of the system through the use of simulation and real robots.
- New approach to represent modular robotic configurations based on the Assembly Incidence Matrix (AIM) method by adding independent modular variables to allow easy synthesis of a wide range of modular robots.

1.4 Organization of the Dissertation

The remaining of this dissertation is organized as follows: Chapter 2 reviews current research on evolutionary robotics and modular robots in order to identify the most promising methods that can lead to designing modular robotic systems autonomously. Chapter 3 describes the task-based optimal modular robotic design system components including the mathematical model, kinematic model, and optimization model. This chapter starts by introducing the conceptual model of the used robotic modules. Then, it presents the mathematical model for enumerating modular robotic assembly configuration. Finally, it explains the use of Evolutionary Algorithm to solve the task-based design optimization problem. Chapter 4 discusses how the previously presented technical aspects are implemented. Chapter 5 presents and analyzes the results. Finally, chapter 6 concludes the dissertation, summarizes the key findings, and recommends future work.

CHAPTER 2: RELATED WORK

Producing autonomous adaptive robots is a huge challenge. In biology, autonomous and adaptive creatures are produced using evolution. However in industry, mainstream robots use machine learning to produce adaptive behavior to simulate biological aspects while neglecting the autonomous side of it. Therefore, evolutionary algorithms are used to optimize robotic autonomy and adaptation producing what is known as evolutionary robots [3].

Evolutionary robotics evolves populations of robots by applying evolutionary computational methods, and then selects the fittest to survive. The benefits of evolutionary algorithms include the power to improve the parameters and the structure of the robotic control and morphology [4-5]. In order to maximize that power, modularity could play a role as the basic building block in the robotic system to simplify the implementation and search space because of its discrete nature [6].

This chapter starts with reviewing some research work done on evolving robotic control in a fixed morphology and co-evolving robotic control and morphology. Then, modular robotics are discussed as a method to design and implement evolutionary robots in the physical world, as advanced technology and rapid prototyping techniques have made these modular robots feasible. Moreover, evolutionary computation can empower modular robots by allowing them to self-assemble, self-reconfigure, self-repair, and self-reproduce. Thereafter, numerous modular robotic applications are analyzed along with their

capabilities of performing various evolutionary challenges. Finally, the current state of the art and challenges are discussed.

2.1 Evolutionary Robotics

In nature, evolution produces heritable changes in the phenotypes of organisms over multiple generations for better adaptation to the environment. In robotics, evolution has been proposed as a nature inspired approach to avoid the bias and limitations introduced by human designers and to produce better adapted robots to the environmental changes [7]. Simply, evolutionary robotics is a method of creating autonomous robots automatically without human intervention [8].

The Darwinian theory of evolution inspired evolutionary robotics. This theory states that all organisms develop through mutation, crossover, and selection that increase the new generation's ability to compete, survive, and reproduce [9]. Based on the principle of selective reproduction of the fittest, robots are viewed as autonomous artificial organisms that can develop their own skills by interacting with the environment and without human intervention. The fittest robots survive and reproduce until a robot that satisfies the performance criteria is produced [10].

Each robot comprises two major parts: control (brain) and morphology (body). Controls are represented in many ways including neural networks that map sensory input to actuator outputs. Morphology can be described as tree-based representation, L-system consisting of set of rules that can produce construction sequences or regulatory networks.

To allow for open-ended synthesis, both control and morphology should co-evolve along with the fitness functions and evaluation methods [11].

Evolving robotic controls has received a lot of attention on research because controls are more adaptable than morphology. Floreano et al. described evolving a small-wheeled robot control that is comprised of a neural network using a simple Genetic Algorithm to navigate a looping maze. Their experiment showed that the fitness function evolved, and the cruising speed of the robot evolved as well, demonstrating that evolution can lead to better adaptation [8]. Nolfi and Floreano presented a set of navigational experiments in their book, ranging from the simple to the very complex in order to address different adaptation mechanisms. In some cases, the evolved solution outperformed the hand-designed solution by capitalizing on interactions between machine and environment that could not be captured by a model-based approach. On the other hand, more complex tasks exposed the limits of reactive architectures [10].

In the previous work, robotic controls were evolved for fixed morphological structures that were user-designed. Other research studies indicate the need to co-evolve the robotic control and morphology in order to produce fitter robots, as is the case of nature. Paul and Bongard introduced coupled evolution of robotic morphology and control on a biped robot in simulation. The closed loop recurrent neural network controller was optimized simultaneously with the morphological parameters using a fixed length Genetic Algorithm [12]. Zykov et al. applied the same theory on a physical robot to evolve the dynamic gates in hardware. The nine-legged robot's open-loop controller was evolved

using a Genetic Algorithm to allow evolving speed and locomotion pattern under the rhythmicity constraint [13]. Lund investigated the co-evolution of robotic controller and morphology using LEGO parts to construct the evolved morphology and downloaded the evolved controller to LEGO MINDSTORM RCX. The search space for morphology was limited, but the solution search space was enlarged when co-evolving controller and morphology [14, 15]. Sims created a system where virtual robotic control and morphology co-evolve to allow the robots to compete in a physically simulated 3D world to gain control over common resources. The robots were made of 3D cubes and oscillators [16].

The techniques presented for coupled evolution of robotic control and morphology allowed the automation of designing complex systems that would be difficult to design using traditional methods. Lipson explored automatic design concepts by building robots using lower-level building blocks with no sensors. The control was composed of neurons and the morphology was composed of bars and linear actuators. The resulting solutions were remarkably elaborate and would have been difficult to design using traditional methods [16]. However, an obvious constraint was the manufacturability of the resulting solutions. Therefore, Faíña et al. proposed the use of modular robots as the fundamental building blocks for evolutionary processes because modularity allows building a wide variety of robotic structures, simplifies the search space, and ensures easy implementation in reality [7].

2.2 Modular Robotics

Modular robots are composed of various units or modules, hence the name. Each module involves actuators, sensors, computational, and communicational capabilities. Usually, these systems are homogeneous where all the modules are identical; however, there could be heterogeneous systems that contain different modules to maximize versatility [7].

Modular robotic systems have three promises: versatility, robustness, and low cost. Versatility is the capability of the modular robotic system to form a number of different shapes, each with big numbers of degrees of freedom (DOF). In other words, to allow the robot to self-reconfigure in order to accomplish various tasks in different environments. Versatility can be measured by the number of isomorphic configurations the robotic system can form and by the number of DOF in the system. The number of configurations grows exponentially with the number of modules and the number of DOF grows linearly with the number of modules. Robustness comes from redundancy and self-repair. When the robot is composed of many identical modules and one fails, any other module can replace it to keep the system running. Finally, low cost promise is achieved through batch fabrication. As the number of repeated modules increases, the economies of scale come into play and the per-module cost goes down [18]. Also, maintaining low cost can be achieved through rapid prototyping equipment techniques; such as 3D printing, that can build any object by laying down successive layers of material. In order to empower the aforementioned

characteristics in modular robotic systems, evolutionary task-based design approaches are utilized to replace traditional design methods.

In 1995, Chen and Burdick proposed a system to find optimal modular robotic design to accomplish a specific task. This system is formulated as a discrete optimization function based on assembly incidence matrix to represent the robotic configuration. These matrixes are encoded into bit strings to best utilize Genetic Algorithm. Genetic Algorithm is used for optimization because of the discrete nature of the search space, even though the application of Genetic Algorithm could be computationally expensive. The robots consist of link and joint modules and were implemented in simulation [19].

Chung et al. introduced in 1997 a task-based design method for modular robot manipulators. The robotic system consists of manipulator base, link, and joint modules. The robot configuration is determined using kinematic relations. Then, Genetic Algorithm is used to find the optimal link length for a specific task. Because of the complexity of the problem, the computation effort required for Genetic Algorithm is tremendous. This algorithm has been implemented on physical robots. This work considers only the first level of modular architecture; which is kinematics synthesis [20].

In the same year, Chocron and Bidaud presented an adaptive multi-chromosome evolutionary algorithm for optimizing task-based kinematic design of modular robotic systems. The robot consists of a mobile base and a set of link and joint modules to assemble the manipulator arm. The task is specified as 3D end-effector configurations. The robots

were implemented in simulation. This method yielded better results compared to two-level GA and multi-chromosome evolutionary algorithm; however, it still lacks optimality. Additionally, the increase in the number of design variables increases the search space exponentially that results in making this algorithm insufficient [21].

According to Yang and Chen, with fewer DOF, the modular robot can better perform the task in terms of energy consumption and loading capacity. Therefore, they proposed the minimized degree-of freedom concept for task-based modular robot design optimization in 2000. This system uses assembly incidence matrix to represent the robot configuration as in [19]. It also uses Evolutionary Algorithm (EA) to search for optimal solutions. This algorithm was implemented in simulation and produced sub-optimal results [22].

Hornby et al. proposed in 2001 an automatic design system for modular physical robots that can become more complex. The goal of their work is to overcome the limitations of using evolutionary design approach to make it ready for practical engineering by reaching high complexities and to simplify design changes by using generative design and allowing reusability of modules. The robots consist of bars and joints inspired by Tinker-Toy™ components. The 2D robots are produced in simulation using Lindenmayer system for design and evolutionary algorithm for optimization. Then the actual robot is hand-assembled from an evolved design. This work described the co-evolution of the robotic control and bodies for locomotion abilities and produced a real robot that was tested and moved in the real world [23].

In 2013, Faíña et al. proposed an evolutionary designer for heterogeneous modular robots. This system is capable of designing complete robots including their control and morphology. Each robot involves distributed control and heterogeneous modular architecture. Tree like representation is used for robotic morphology to smooth the search space. A constructive evolutionary strategy is used to co-evolve the robots control and morphology in simulation. Then the resulting robots are implemented physically to prove feasibility. The results were promising when applied to solve a linear motion problem but needed improvements in the case of static problems [7].

Moreover, evolutionary algorithms can be applied to modular robots to allow self-assembly from constituent modules, self-reconfiguration into different functional forms, self-repair to detect errors and recover from failures, and self-reproduce where one system can produce another autonomous functional system.

2.2.1 Self-Assembly

One of the main benefits of modularity is the capability of self-assembly, which is the natural construction of complex multi-unit system using simple units governed by a set of rules. The self-assembly process is ubiquitous in nature as it generates much of the living cell functionality [24]. However, it is uncommon in the technical field because it is considered as a new concept relatively in that arena, although it could help in lowering costs and improving versatility and robustness; which are the three promises of modular

robotics. The ability to form a larger stronger robot using smaller modules allows self-assembled robots to perform tasks in remote and hazardous environments [25].

Various types of evolutionary algorithms have been utilized in implementing self-assembly modular robotic systems. Bonabeau et al. studied employing Genetic Algorithm to generate self-assembly rules for modular robotic systems and explored the relationship between the space of possible rules and resulting biologically plausible structured architectures. This work did not address the problem of self-assembly according to pre-determined shapes [25].

Tolley et al. extended the stochastic self-assembly modular robot proposed in [26] from 2D to 3D. They used evolutionary approach to design robotic structures according to an input function. These structures are evolved in simulation using frequency-based representation. Then the assembly algorithm takes place to plan the assembly of the fittest evolved robot by sampling a graph of all possible paths to the target structure and following those that leave the most options open. The modules in this system are unable to move on their own because they need to circulate in turbulent fluid to accrete onto the structure. This fluidic system could be scaled down to produce micro-scale modules. This system lacks the possible feedback between the design and assembly phases, which can have a large impact on the evolving design if implemented to adapt to assembly conditions [27].

2.2.2 Self-Reconfiguration

Recently, modular robotics has gotten attention from researchers in the robotics field because of their ability to self-reconfigure [28]. Modular self-reconfigurable robots involve various modules that can combine themselves autonomously into meta-modules that are capable of performing various tasks under different circumstances [7]. The ability to self-reconfigure allows these robots of metamorphosis, which in turn makes them capable of performing different sorts of kinematics. For instance, a robot may reconfigure into a manipulator, a crawler, or a legged one [28]. This sort of adaptability enables self-reconfigurable robots to accomplish tasks in unstructured environments; such as space exploration, deep sea applications, rescue missions, or reconnaissance [29].

Yim et al. in 2002 classified reconfigurable robots into three classes of architecture: lattice, chain, and mobile based on how they reconfigure [29]. Then they added deterministic and stochastic reconfigurations in 2007 [30].

Lattice architectures have modules that are arranged in a 2D or 3D grid that can be used as a guide for modules to determine their positions and form the new shape accordingly. All modules remain attached to the main body to simplify planning and control [30]. One example of a lattice-based self-reconfigurable robot is M-TRAN [32-33].

Chain/Tree architectures have modules that are connected together in a string or tree topology. The serial underlying architecture implies that each chain is always attached

to the rest of the modules at one or more points, and the modules reconfigure by attaching and detaching to and from themselves [30]. Chain architectures are more versatile compared to other architectures due to their capability of reaching any point in space through articulation, but they are more difficult to control and more expensive computationally to represent and analyze [28]. An example of a chain-based self-reconfigurable robot is PolyBot [34-36].

It is important to mention that lattice architecture and chain architecture do not contradict, and numerous systems can be of both types at the same time, such as SuperBot [37] and UBot [38]. These systems tend to have Hybrid architectures [29].

Mobile architectures have modules detach from the main body and maneuver independently using the environment; e.g. liquid or outer space, to link up at new locations in order to form new shapes, complex chains or lattices, or form a number of smaller robots. Mobile architecture is less explored compared to other structures because the reconfiguration difficulty outweighs the functionality gain [29-30]. A mobile-based self-reconfigurable example system is CEBOT [39-41].

Deterministic Architectures have modules move directly to their target locations during the self-reconfiguration process. Each unit's location can be known at all times or calculated at run time, such that reconfiguration times are guaranteed. Feedback control is necessary to ensure precise movement. Usually, macro-scale systems are considered deterministic [30].

Stochastic Architectures have modules move in a 2D or 3D environment using statistical processes; e.g. Brownian motion, which are used to guarantee reconfiguration times. The exact location of each unit is known only when it is connected to the main structure, but the paths taken by those units to move between locations might be unknown. Stochastic architectures are more ideal at micro-scale systems [30].

Evolutionary algorithms were used to evolve modular self-reconfigurable robotic controls in order to support self-reconfiguration and also to implement different modular robotic behaviors. Østergaard and Lund explored evolving controllers of M-TRAN [42] and ATRON [31] self-reconfigurable modular robotic systems in simulation. Employing Genetic Algorithm for implementing M-TRAN walking behavior is very complicated because evolving each controller locally to generate a global behavior is affected by the conditions of neighboring modules. Therefore, when attempting to evolve ATRON controllers individually to allow the modular collection of moving in the right direction, two modules were evaluated as a couple instead of evaluating single modules using competitive co-evolution and symbiotic co-evolution. This work did not address the constraints of physical systems [31].

ACMoD is a modular self-reconfigurable robot that uses Genetic Algorithm to produce proper configuration patterns and for optimizing the path of modules through a static grid of different terrain blocks. This work did not address dynamic environment or found optimal solutions. The system was implemented in simulation [43].

2.2.3 Self-Repair

Self-repair is a special type of self-reconfiguration that allows a robot to replace damaged modules with functional ones in order to continue with the task at hand [28]. A self-repair system must have two qualities: the ability to self-modify, and the availability of new parts or resources to fix broken ones. Therefore, modular self-repair robots usually consist of redundant modules. Self-repair involves two phases: detecting the failure module, and then ejecting the deficient module and replacing it with an efficient extra module. Such robots are well suited for working in unknown and remote environments.

ATRON modular robot uses evolutionary algorithms to implement self-repair functionality [31]. This system is discussed in detail in the Applications section in this chapter.

2.2.4 Self-Reproduction

The ultimate form of self-repair is self-reproduction; which allows robots to reproduce themselves from an infinite supply of parts using simple rules. If the resulting system is an exact replica of the original, the system is called a self-replicator [44]. The effort in self-reproducing is focused on the design and construction of a small seed system that grows exponentially to form a larger system through tens of generations. The resulting self-reproducible robots are capable of accomplishing very large-scale tasks, such as collection of solar energy, direct removal of greenhouse gases from the Earth's atmosphere, and water purification for irrigation. Self-reproduction differs from self-

assembly because the resulting systems do not need to make copies of themselves in the latter cases. Since any replication process requires an external material supply, some lattice positions may act as dispensers, where new modules reappear when removed from that location.

Self-replicators can use evolutionary algorithms to evolve into the goal structure. The evolution occurs in two stages: morphology evolution and control evolution. Zykov et al. used Genetic Algorithm to evolve Molecube in 2D simulation using two distinct fitness functions: one for evaluating the fitness of morphology and the other to evaluate the fitness of control. The robotic structure was expressed using a variable-length genome and the control was described using command sequence. Only few results were successful yielding separate identical copy and a matching control. The successful results were implemented on physical robots. This work faces a computational challenge in the planning of self-replication algorithms [44].

2.3 Applications

There is a growing number of modular robotic prototypes that has been studied in the literature, so this section reviews a number of emphasized prototypes that participated in the growth of evolutionary modular robotics research.

The timeline covered in this paper ranges from 1990 until this year. Figure 2.1 illustrates a chronogram of some of the surveyed systems along with other systems that

were not covered in detail in this paper. Tables 2.1-2.3 compare those systems based on different parameters.

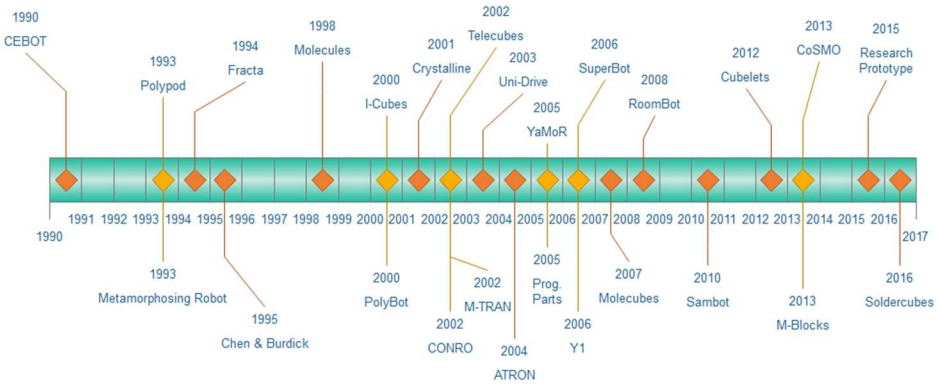


Figure 2.1. Chronogram of selected modular robotic prototypes

Table 2.1. Modular robotic systems classification based on holistic system characteristics

	Self-Assembly	***Self-Reconfiguration***	***Self-Repair***	***Self-Replicate***
PolyBot		√		
I-Cubes		√		
Crystalline		√	√	
Telecubes		√		
CONRO		√		
M-TRAN		√		
ATRON		√	√	
Prog. Parts	√			
YaMoR		√		
Y1				
SuperBot		√		
Molecubes		√		√
RoomBot	√	√		
Sambot	√	√		
Cubelets				
M-Blocks	√	√		
CoSMO		√		

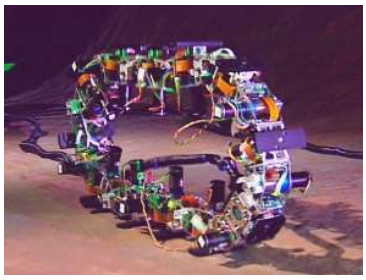
Table 2.2. Modular robotic systems classification based on modularity state of matter

	Homogeneous	***Heterogeneous***
PolyBot	√	
I-Cubes		√
Crystalline	√	
Telecubes	√	
CONRO	√	
M-TRAN	√	
ATRON	√	
Prog. Parts	√	
YaMor	√	
Y1	√	
SuperBot	√	
Molecubes	√	
RoomBot	√	
Sambot	√	
Cubelets	√	
M-Blocks	√	
CoSMO	√	

Table 2.3. Modular robotic systems classification based on implementation method

	Simulation	***Physical Implementation***
CEBOT	√	
Polypod		√
Metamorphosing Robot	√	
Fracta		√
Chen & Burdick Robot	√	
Molecules		√
PolyBot		√
I-Cubes		√
Crystalline		√
Telecubes		√
CONRO		√
M-TRAN		√
Uni-Drive		√
ATRON		√
Prog. Parts		√
YaMor		√

	Simulation	*Physical Implementation*
Y1		√
SuperBot		√
Molecubes		√
RoomBot		√
Sambot		√
Cubelets		√
M-Blocks		√
CoSMo		√



(a)

(b)

(c)

(d)

(e)

Figure 2.2. (a) PolyBot G2 [35] (b) M-TRAN III [48] (c) ATRON [31] (d) Programmable Parts [50] (e) Molecubes [54]

2.3.1 PolyBot – 2000

PolyBot is a modular self-reconfigurable robot that was implemented to explore how realistic it is to make robots using several homogeneous hardware modules. Three generations of PolyBot modules were prototyped, such that each generation addresses a number of shortcomings discovered in the previous one. The first generation (G1) is constructed using two modular types: node and segment. Unlike its G1 predecessor, the second generation (G2) connection ports have electromechanical latches under software control. These latch onto the pins protruding from the opposite face. The third generation (G3) modules are smaller and lack the DC motor extending past the side of each module. The new module has instead a DC pancake motor with a harmonic gear that is completely internal. The connectors are larger and have higher contact force for higher current loads to enhance performance [34-37].

PolyBot is capable of self-reconfiguration by changing its geometry and locomotion mode depending on the terrain type. Planning the self-collision-free motions can be challenging because the size of this space is exponential in the number of modules, but proportional to the number of DOF. For many applications, a fixed set of configurations is sufficient. In this case, reconfigurations can be pre-planned off-line and stored in a Table for ease of reconfiguration [17].

2.3.2 Telecubes – 2002

Telecubes are cubic modules that were introduced by Suh et al. as an extension to the Crystalline system [45]. Each cube has six prismatic DOF and sides capable of

expanding more than twice its original length. These cubes can form a modular self-reconfigurable robot by attaching and detaching magnetically to other cubes [46]. However, the reconfiguration algorithm lacked local decision making and parallel execution [47].

2.3.3 M-TRAN – 2002

M-TRAN (Modular Transformer) is a distributed lattice-based self-reconfigurable robotic system that can metamorphose into various configurations, such as a legged machine generating walking motion. In order to drive M-TRAN hardware, a series of software programs has been developed including a kinematics simulator, a user interface for designing configurations and motion sequences, and an automatic motion planner [32].

M-TRAN II is the second prototype where many improvements took place to allow versatile whole-body motions and complicated reconfigurations. Those improvements contain a reliable attachment/detachment mechanism, high-speed inter-module communication, on-board multi-computers, accurate motor control, and low energy consumption. The software has been improved as well to verify motions in dynamics simulation and to design self-reconfiguration processes [33].

The third prototype, M-TRAN III, has been developed with an improved connection mechanism. Various control modes including single-master, globally synchronous control and parallel asynchronous control are made possible by using distributed control. Self-reconfiguration experiments using up to 24 units were performed

by centralized and decentralized control. System scalability and homogeneity were maintained in all experiments [48].

2.3.4 ATRON – 2004

Another modular self-reconfigurable robot is ATRON, a lattice-based system consisting of approximately spherical modules, where each sphere is constructed as two hemi-spheres joined by an infinite revolute joint. Actuation is realized as rotation around an axis diagonally through the sphere. This design allows for a very stable construction around the actuated joint since a relatively large area is available for mechanics. However, the spherical basic module design makes it hard to have large flat surfaces connecting to each other. With spherical modules, connectors need to establish essential point-to-point contacts between modules, which are not desirable because of the high collision probability. The limited mobility of ATRON along with other motion restrictions leads to the use of ATRON meta-module to reduce motion constraints. The meta-module is composed of three modules: a body in the center that is connected to two legs.

Modular ATRON control comprises Artificial Neural Networks. Genetic Algorithm is used to optimize the weights of the ANNs. Even though ATRON modules are minimalistic because they have only one actuated DOF, a group of modules was capable of self-reconfiguring in 3D simulation. Similarly, ATRON modules demonstrated self-repair successfully in simulation [31, 49].

2.3.5 Programmable Parts– 2005

In 2005, Bishop et al. built triangular programmable parts that can be assorted on an air table by overhead oscillating fans to self-assemble into various shapes according to the mathematics of graph grammars. The modules can communicate and selectively bond using mechanically driven magnets, without global knowledge of the full shape. Despite planning to build approximately 100 parts, only six parts were built for design simplicity reasons. Those six parts were used in an experiment that showed how these parts react similarly to chemical systems [50]. In addition, Napp et al. provided kinetic rate data measurements to the previous work of graph grammar in order to yield a Markov Process Model [51].

2.3.6 Molecubes – 2007

Molecubes system is an open hardware and software platform for modular robotics that was developed to remove entry barriers to the field and to accelerate progress. The system is composed of modules with one rotational DOF. Different types of active modules, such as gripper, actuated joint, controller, camera, and wheel along with a number of passive modules were presented. Each module is a cube shaped with round corners that comprises approximately two triangular pyramidal halves connected with their bases so that their main axes are coincident. Each of the six faces of the module is equipped with an electromechanical connector that can be used to join two modules together. Symmetric connector design allows four possible relative orientations of two connected module interfaces, each resulting in different robot kinematics [52].

Genetic Algorithm is used to evolve the modular neural network control of the robots in simulation to generate a certain behavior or motion [53]. In order to achieve self-replication, path planning is done with a gene pool that has been built using evolutionary algorithm [54].

2.3.7 iMobot – 2010

Ryland and Cheng designed an intelligent self-reconfigurable modular robot with each module having 4 DOF and 6 connection faces. In this robotic system, the individual modules have full mobility unlike all the other systems discussed in this paper where the modules must be connected in a cluster to perform all types of locomotion. For example, M-TRAN module can crawl, but it needs a second module to turn. In addition, the iMobot robotic system can perform unique locomotion modes such as driving and lifting into a camera platform [55]. iMobot uses a distributed agent based Genetic Algorithm to search for the optimal genotype that can generate a certain robotic gait [56].

2.3.8 UBot – 2011

UBot is a modular self-reconfigurable robotic system that is capable of multimode locomotion. The locomotion modes include cross, loop, quadruped and other gaits. UBot is a hybrid system combining the advantages of lattice and chain self-reconfigurable robots. Each module is cube shaped based on one universal joint and has two rotational joints in order to achieve 2 DOF [38, 57].

In 2013, a 3D dynamic simulator was developed to simulate rigid body dynamics and evolve robotic locomotion. Evolutionary Robotics was used to find a gait planner for different robotic structures [58].

2.3.9 SMORES – 2012

Self-assembling MOdular Robot for Extreme Shape-shifting (SMORES) was designed to become a universal modular robot that improves the versatility of self-reconfigurable robots by allowing the robots to reconfigure in three reconfiguration classes; lattice, chain, and mobile. Each module has four active rotational DOF and two wheels to allow mobile movement [59].

A design framework was developed in 2018 to facilitate configuration design of SMORES robots. The proposed system verifies robotic design validity by detecting conflicting commands and loss of stability. Moreover, this system allows existing robotic structures reusability to create complex robotic designs and behaviors [60].

2.4 Current State of the Art

More recently, new efforts have been pursued in the field of evolutionary modular robotics. Many tasks have been shown to be achievable, especially with the high number of physically implemented robotic systems. However, the majority of modular robotic behaviors; such as self-reconfiguration and self-repair were implemented in simulation despite the existence of a physical prototype; which can be considered as a reality gap. The reason behind this is the high cost of performing evolution inside physical hardware

because of power, communication, and other limitations [61]. Researchers have paid attention to bridge the reality gap using rapid prototyping techniques, as this fabrication method becoming more accessible [6].

Auerbach et al. tried to bridge the reality gap by proposing RoboGen, an open-source software and hardware platform that 3D prints evolved modular robots. The software component of RoboGen contains an evolutionary engine to produce modular robots and a simulator to evaluate the fitness of the evolved robots. The robot morphologies are represented as genetic programming trees and the controls are represented as artificial neural networks. Then, the evolved robots can be manufactured using desktop 3Dprinters; as illustrated in Figure. 2.3 [62].

Additive manufacturing was used by Samuelsen and Glette to build an autonomous robotic system that automatically designs and generates modular robots in simulation using evolutionary algorithms. Then, the system uses an automatic cluster to select the robots with highest fitness values to be manufactured using off-the-shelf motors and 3D printed structural components. The same evaluation procedure was used to compare the performance of the simulated robots and the performance of the physical robots. Five robots were manufactured and three out of five robots have significantly lower performance in reality [63].

Finally, Cellucci et al. demonstrated a 1D printing system inspired by the ribosome to automatically design and fabricate various robots using the same source material. Although the resulting robots have modest functionality, this research points towards

expandable robotics that rapidly fabricate customized robots on demand and allow robots recycling to produce new robotic designs that can perform new tasks [64].

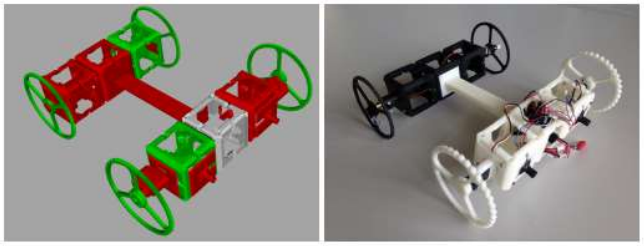


Figure 2.3. Evolved robot: simulation (left) and reality (right) [62]

2.5 Conclusion

This chapter has surveyed different applications of evolutionary algorithms in evolving robotic control systems, co-evolving robotic control and morphology, and evolutionary task-based design of modular robots. It was found that evolutionary based solutions exceeded hand designed ones by showcasing novel characteristics and capabilities. The use of evolutionary algorithms in the modular robotic field allowed self-assembly, self-reconfiguration, self-repair, and self- reproduction. These systems were discussed to express how evolutionary robotics can be used to generate rules for planning and controlling general robotic behaviors. Then numerous modular robotic prototypes were analyzed to demonstrate the feasibility of evolutionary modular robotic systems that are capable of accomplishing various tasks in dynamic environments.

CHAPTER 3: TASK-BASED OPTIMAL MODULAR ROBOTIC DESIGN

Modular robotic systems consist of collections of units called modules that can be assembled to form various robotic configurations that are capable of performing different tasks and adapting to diverse environments by reconfiguring to different shapes. These dynamics granted modular robotic systems high versatility, robustness, and low production cost compared to mainstream industrial robots that are designed to perform general tasks. However, one of the main challenges facing adopting modular robots is the problem of designing an optimal robotic system to accomplish a certain task when given only the task to be achieved and an inventory of available modules. Therefore, we propose a solution for the design problem as a task-based optimization model for generating optimal modular robotic designs that meet the task requirements, as demonstrated in Figure 3.1.

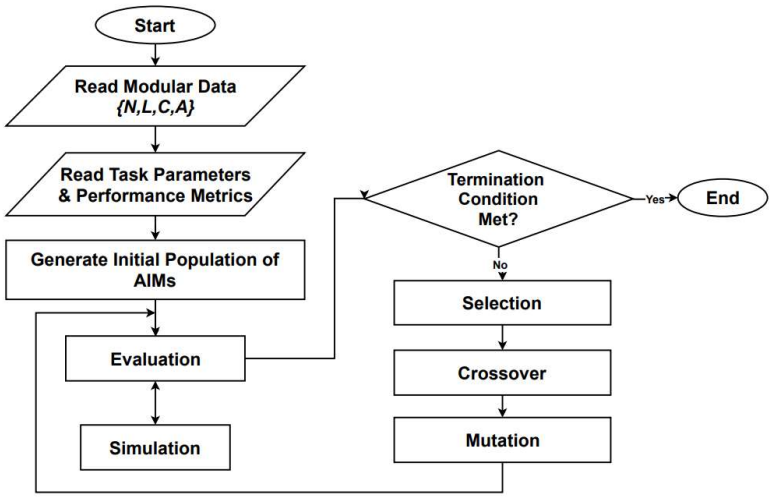


Figure 3.1. Task-Based Optimization System of Modular Robotic Design

The proposed system takes as input the set of modules in the available modular inventory; as well as the input task to be performed by the resulting robot. The system consists of a design optimization model that uses Genetic Algorithm to find the task-based optimal modular robotic designs. The resulting robots are evaluated automatically in simulation and the best designed robots are hand assembled to represent the output of the system.

The modular parameters of a robot can be described using the proposed kinematic model and the resulting robot configuration assembly can be expressed using Assembly Incidence Matrix (AIM). All of the aforementioned concepts are covered in Section 3.1. Section 3.2 discusses the input robotic task specification and how it should be formulated to generate the best possible results. The design optimization model components are introduced in Section 3.3. Then, the evolutionary algorithm used is explained in Section 3.4.

3.1 Robotic Modeling

3.1.1 Robot Modules

The robotic modules can be of any shape such as hexagonal, square, cube, oval, or triangular; as shown in Figure 3.2. For instance, 2D hexagonal modules were used to build Fracta [65] and Metamorphosing robot [66]. Two dimensional square modules were used to build crystalline [44] and 3D cube modules were used in numerous robotic systems including Polybot [16], M-TRAN [32], Telecubes [45], and Cubelets [68] because of their

simplicity when it comes to implementing the individual modules and the overall robotic system.

In this work, Dtto modular robotic kit is used to build the robot in simulation and reality. The shape of a Dtto module is considered as a special class of 3D cube shaped modules, where each single module is composed of two connected modules. The reason for choosing this modular design is to allow every single module of moving independently without the need to connect to other modules. Although a single unit can move independently, it still needs to connect to another "helper" module in order to turn. Each cube shaped module has 6 connection faces and 6 actuated DOFs, so the modules can attach magnetically in different orientations to form more complex robotic shapes; as demonstrated in Figure 3.3 [69].

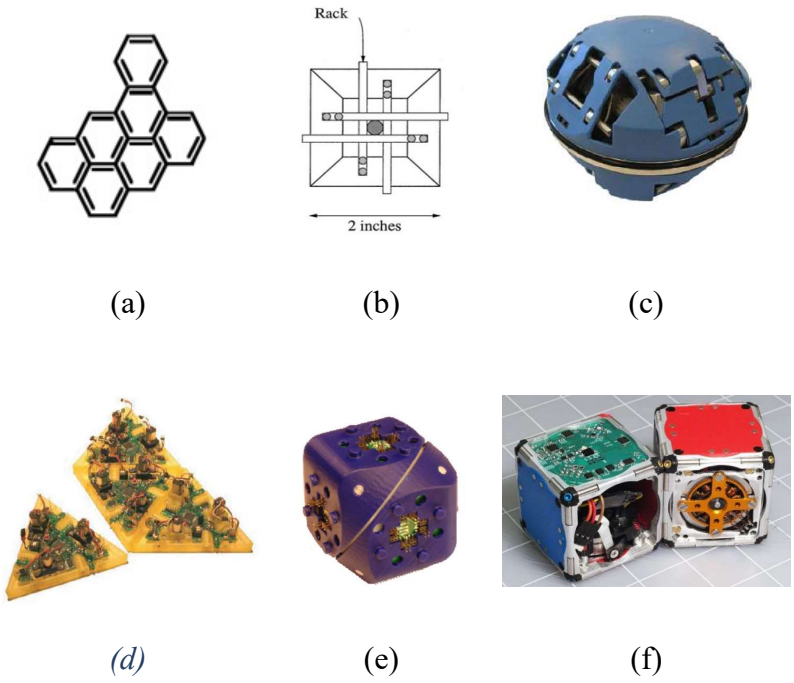
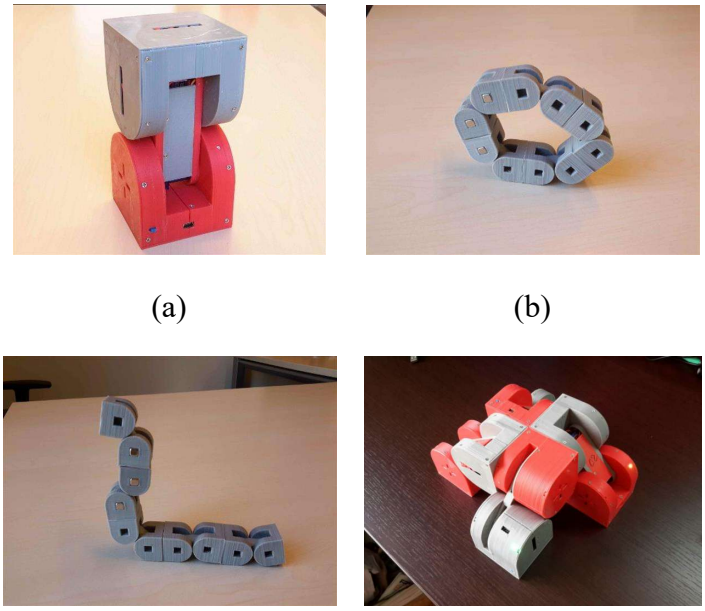


Figure 3.2. (a) 2D hexagonal modules of Metamorphosing Robots [66] (b) 2D square modules of Crystalline [44] (c) 3D semi-spherical modules of ATRON [31] (d) 3D triangular modules of Programmable Parts [24] (e) 3D cubic modules of Molecubes [52] (d) 3D cubic modules of M-Blocks [70]



(a) (b)

(c) (d)

Figure 3.3. (a) Single Dtto module (b) Wheel robot (c) Snake robot (d) Walker robot. All robots are made of Dtto modules

Table 3.1. Specifications of Dtto

Item	***Value***
Dimensions	64 x 64 x 130 mm
Weight	210 g
CPU	Arduino Nano v 3.0
Main Movement	2 TowerPro SG92R Servomotor (2.5kg/cm)
Coupling Mechanism	3 TowerPro SG90 Servomotor (1.8kg/cm)
Connection Faces	24 Neodymium Disk Magnets (4x3 mm)
Communication	Bluetooth HC-06 NRF24L01+ Wireless Transceiver
Power Supply	2 Li-Po Battery 3,7V 600mAh 25C

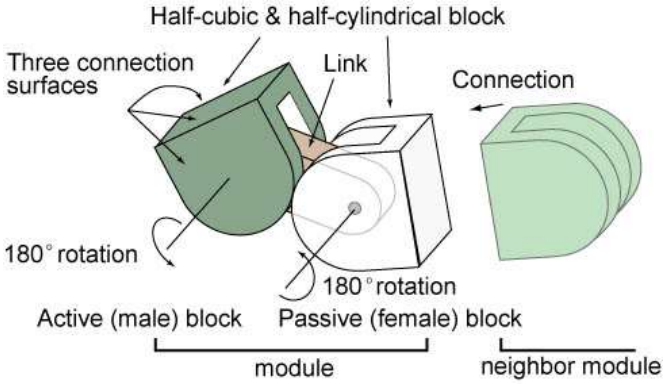


Figure 3.4. Schematic View of Dtto Module

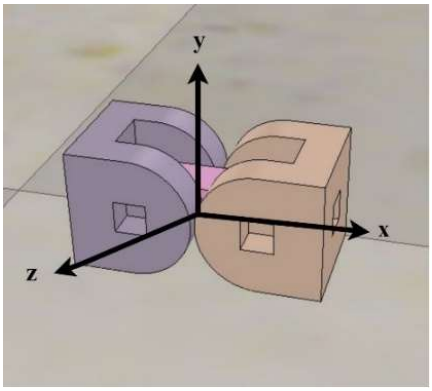


Figure 3.5. Kinematic Model

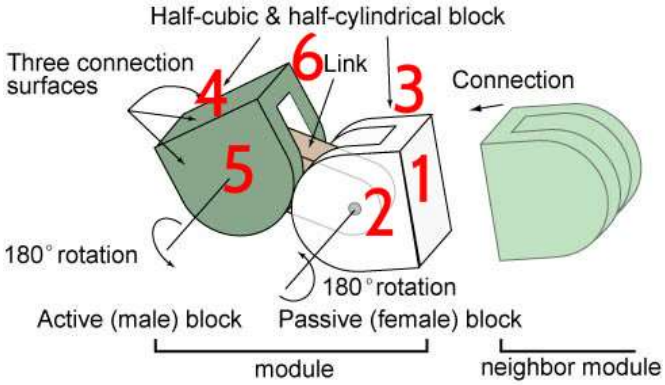


Figure 3.6. Labels of Connection Sockets

Dtto modular self-reconfigurable robot is used in this research to evaluate the evolved robots in simulation and build the physical resulting robots. Dtto is composed of double-cube modules inspired by M-TRAN III modular design [48]. Every module comprises two identical semi-cylindrical boxes connected by a link, such that each box can rotate 180˚ independently around that link; as shown in Figure 3.4. The single Dtto module has 6 connection faces; thus each box has 3 connection faces of opposite polarities – one block is active (male) while the other one is passive (female). Each module includes actuators, sensors, micro-processors, and electro-magnetic connectors. The modular specifications are listed in Table 3.1.

3.1.2 Kinematic Model

In order to facilitate autonomous robotic assembly and reconfiguration, a kinematic model is proposed to express modular parameters of the robot; i.e. the type and sequence of the modules. This model considers modules as link modules and connection faces as connection sockets, thus female connection sockets of a certain link module can be directly connected to male connection sockets of another link module. The link module set L contains the available link modules to be used in the task-based design framework.

In this kinematic model, each link module has a designated body coordinate system originating from the center of that link module, as shown in Figure 3.5. The z-axis is perpendicular to the link connecting the semi-cylindrical boxes of the link module and the x and y axes are pointing towards the connection sockets. This coordinate system is useful for labeling the faces of the link modules, because the orientation of every connected link

modules in a robotic structure can affect the resulting locomotion gait of that robot. The labeling of the faces starts with 1 goes on until 6 as shown in Figure 3.6.

3.1.3 Modular Configuration Representation

Numerous modular robotic assembly configurations can be described using Assembly Incidence Matrix (AIM), proposed by Chen and Burdick in [71], which resembles incidence matrix in modeling how link modules are connected and showing connecting sockets information.

In order to represent a modular robot using AIM, we first need to describe the structural topology of that robot using a labeled kinematic graph, where each vertex represents a link module and each edge represents how two vertices are connected.

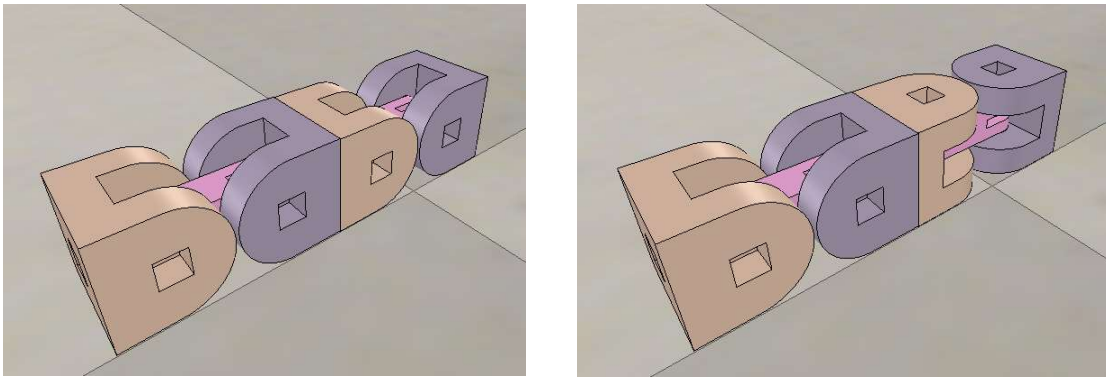
The incidence matrix of a modular robot composed of n modules is composed of n rows (vertices) and $n-1$ columns (edges) can be describes as follows:

$$\begin{array}{c} \\ v_1 \\ \vdots \\ v_n \end{array} \begin{array}{c} \begin{array}{ccc} e_1 & \cdots & e_{n-1} \end{array} \\ \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \end{array}$$

Every entry of 1 demonstrates a connection between the link modules represented by the vertices. Zero entries illustrate that there is no connection. Then, the AIM of that graph is obtained by replacing every entry of one in the incidence matrix of the graph by the label of the associated connecting socket and keeping the zero entries unchanged:

$$\begin{array}{c} \\ v_1 \\ \vdots \\ v_n \end{array} \begin{array}{c} \begin{array}{ccc} e_1 & \cdots & e_{n-1} \end{array} \\ \begin{bmatrix} CF & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \end{array}$$

CF in the previous matrix represents the label of the connection socket.



(a) (b)

Figure 3.7. (a) Dtto robot composed of 2 link modules (b) Another Dtto robot composed of 2 link modules

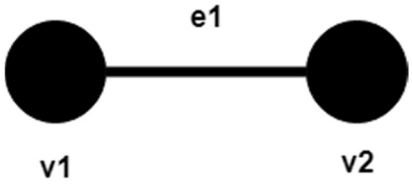


Figure 3.8. Kinematic Graph of the robots illustrated in the above Figure

For instance, the robot illustrated in Figure 3.7 (a) is a modular robot consisting of two Dtto link modules. The associated labeled kinematic graph, G_a, that contains two

vertices and one edges is demonstrated in Figure 3.8. The incidence matrix for this graph is

$$M(G_a) = \begin{matrix} \\ v_1 \\ v_2 \end{matrix} \begin{matrix} e_1 \\ \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{matrix}$$

The AIM for this graph is

$$A(G_a) = \begin{matrix} \\ v_1 \\ v_2 \end{matrix} \begin{matrix} e_1 \\ \begin{bmatrix} 1 \\ 4 \end{bmatrix} \end{matrix}$$

The robot displayed in Figure 3.7 (b) is different that the robot shown in Figure 3.7 (a) because of the orientation of the second link module. However, both robots have the same graph and the same AIM. Therefore, AIM must be modified in order to include more information that can differentiate between the two robots.

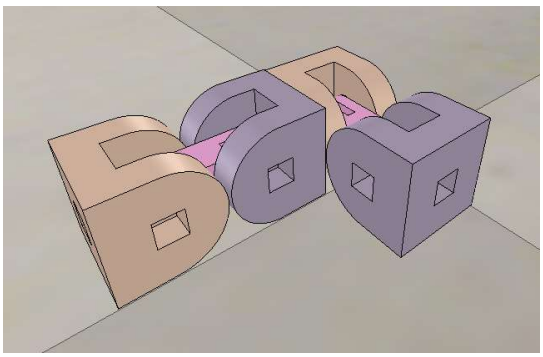
There are different variations of AIM; the extended AIM (eAIM) introduced in [71] can be used for heterogeneous modular robots because it adds an extra column for the types of link modules and an extra row for the types of joints to the original AIM. Another variation used in [72] replaces every connecting socket label in AIM with the physical location of the ports on the link module. This physical location can be described using the coordinate system that belongs to the modular kinematic model described earlier. In this research, AIM is extended by adding to the original AIM, the orientation of the module link in the 3D world relative to its own frame; such that each edge is described by the label of the connection socket followed by the rotation angles around the x, y, and z axis. The rotation angles could have the values of 0˚, 90˚,180˚, or 270˚ and the rotation is performed relative to the frame of the module in the clockwise direction. The modified AIM for G_a is

$$\tilde{A}(G_a) = \begin{matrix} \\ v_1 \\ v_2 \end{matrix} \begin{matrix} e_1 \\ \begin{bmatrix} (1,0,0,0) \\ (4,0,0,0) \end{bmatrix} \end{matrix}$$

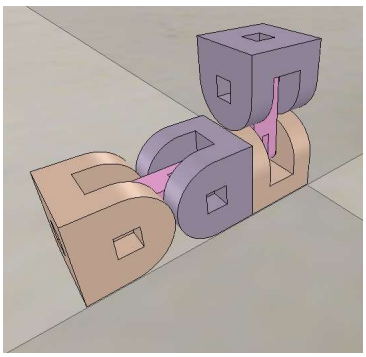
While, the modified AIM for G_b is

$$\tilde{A}(G_b) = \begin{matrix} \\ v_1 \\ v_2 \end{matrix} \begin{matrix} e_1 \\ \begin{bmatrix} (1,0,0,0) \\ (4,90,0,0) \end{bmatrix} \end{matrix}$$

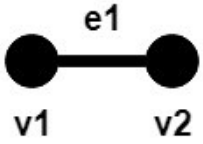
The modified AIM is referenced to as AIM hereunder for simplicity. Two more examples are demonstrated in Figure 3.9 along with their kinematic graph and AIMs. Both robots have the same kinematic graph as the previously discussed robots in Figure 3.7, but different AIMs. Figure 3.10 illustrates relatively larger robot that is composed of three modules along with its kinematic graph and AIM.



(a)



(b)



(c)

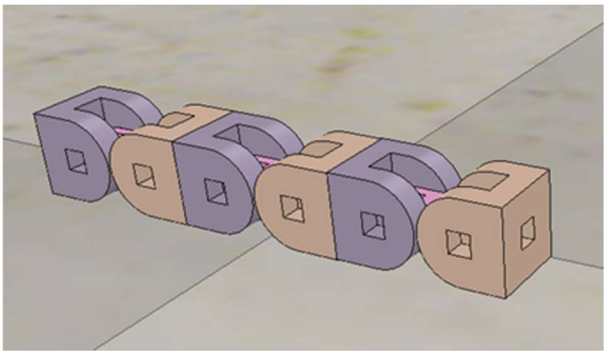
$$\tilde{A}(G_a) = \begin{matrix} \\ v_1 \\ v_2 \end{matrix} \begin{matrix} e_1 \\ \begin{bmatrix} (1,0,0,0) \\ (6,0,90,0) \end{bmatrix} \end{matrix}$$

(d)

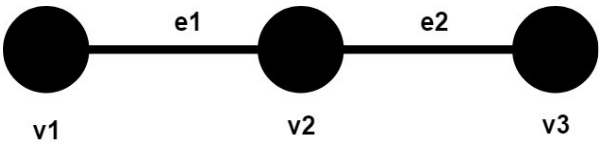
$$\tilde{A}(G_b) = \begin{matrix} \\ v_1 \\ v_2 \end{matrix} \begin{matrix} e_1 \\ \begin{bmatrix} (1,0,0,0) \\ (5,90,90,0) \end{bmatrix} \end{matrix}$$

(e)

Figure 3.9. (a) Dtto modular robot (b) Different Dtto robot (c) Kinematic graph of both robots (d) AIM of robot a (d) AIM of robot b



(a)



(b)

$$\tilde{A}(G_c) = \begin{matrix} \\ v_1 \\ v_2 \\ v_3 \end{matrix} \begin{matrix} e_1 & e_2 \\ \begin{bmatrix} (1,0,0,0) & 0 \\ (4,0,0,0) & (1,0,0,0) \\ 0 & (4,0,0,0) \end{bmatrix} \end{matrix}$$

(c)

Figure 3.10. (a) Dtto modular robot (b) Kinematic graph of the robot (d) AIM of the robot

3.2 Robotic Task Specification

In this work, input tasks are identified as desired motion trajectories describing path planning tasks and containing the start point, the end point, and the sequence of points in between. Each point is described using the xyz coordinate system. Trajectory plans takes into consideration robotic parameters; such as velocity and acceleration. Figure 3.11 illustrates a sample motion trajectory. Another sample trajectory is demonstrated in Table 3.2 as a group of task points and corresponding timestamps.

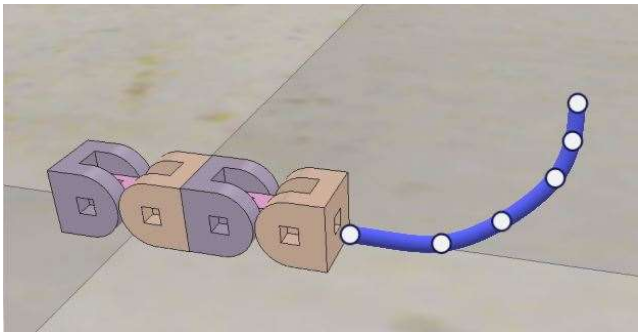


Figure 3.11. Motion trajectory for Dtto modular robot

Table 3.2. Sample trajectory task points and time stamps

	Task Points	***Time Stamp***
1	(3.0, 0.5, 0.5)	t_1=0
2	(3.0, 0.5, 1.5)	t_2=4
3	(2.0, 2.0, 1.5)	t_3=8
4	(2.0, 2.0, 0.5)	t_4=12
5	(0.5, 3.0, 0.5)	t_5=16
6	(0.5, 3.0, 1.5)	t_6=20

3.3 Task-Based Design Optimization Model

An optimization model is formulated to solve the task optimal modular robotic assembly configuration problem. This model includes design parameters, fitness function, and constraints.

3.3.1 Design Parameters

Input module set includes independent parameters that affect modular robotic design. These parameters are:

N — the number of link modules,

L — the type of link modules,

C — the labels of the connection sockets of consecutive link modules, and

$A \mid A \in \{A_x, A_y, A_z\}$ — the rotation angles around the x, y, and z axis respectively.

Thus, the set of design parameters X is defined as:

$$X \rightarrow \{N, L, C, A\}$$

Each robot is designed uniquely when all the above parameters are identified. For every possible robotic deign (N, L, C, A), there exists a unique AIM. The search space includes all possible AIMs.

3.3.2 Fitness Function

Each robotic assembly configuration is evaluated using a task-specific aggregate fitness function that measures the success rate of the task completed by the designed robot. The aggregate fitness function was selected over other types of fitness functions because it can reduce human bias injected into the evolving modular robotic structures [73]. The goal of evaluating every single robotic individual is to examine the suitability of this solution to accomplish a specific task. The fitness function f is defined by listing the essential performance measures and then selecting the most important criterion or combining multiple criteria in a weighted sum function. The rest of performance measures that are not considered in the fitness function can be addressed as constraints.

The general task-specific fitness function can be expressed as follows to measure a certain task outcome:

$$f = \begin{cases} x, & if\ f \leq x_{th} \\ x + \frac{f_{max}}{N}(N - n), & if\ f > x_{th} \end{cases} \tag{3.1}$$

Where x is the expected outcome of the task performed by the designed robot, x_{th} is the threshold value, f_{max} is the fitness of a modular robot with zero modules, N is the maximum number of modules allowed, and n is the number of modules of the individual that is being evaluated.

For instance, in the case of locomotion tasks where the modular robots evolve to travel in an environment, the fitness function of a modular robotic task of moving forward as far as possible on a flat surface is defined as follows:

$$f = \begin{cases} distance, & if\ f \leq distance_{th} \\ distance + \frac{f_{max}}{N}(N - n), & if\ f > distance_{th} \end{cases} \quad (3.2)$$

Where distance is the distance travelled by the robot, $distance_{th}$ is the distance threshold value, f_{max} is the distance travelled by a modular robot with zero modules, N is the maximum number of modules allowed, and n is the number of modules of the current robot.

3.3.3 Constraints

Performance constraints are restrictions defined to ensure the feasibility of a robot configuration while performing a given task. In this research, the majority of considered constraints are mechanical in order to check the mechanical feasibility of a robot assembly. Furthermore, each robot configuration must have a minimum of two link modules in order to be capable of performing the full set of locomotion primitives; moving and turning. Therefore, the following pre-determined structure is used; such that every AIM must contain one base link module:

$$Base - Link - Link - \cdots - Link$$

This pre-determined structure has an effect on restricting the search space by avoiding mechanically infeasible structures [74].

The constraints can include connectivity constraints, collision constraints, and torque limit constraints. The connectivity constraint ensures that all modules must be connected, such that two or more link modules cannot be attached to the same connecting

socket of a certain link module. The collision constraint ensures there is no collision between modules:

$$Collision: Integral = 0$$

The torque limit constraint confirms that a single module can lift a certain number of modules. In our research, the maximum torque of each modular motor is 19.8 kg cm which is enough to support and to connect to three other modules:

$$Torque\ Limit: Maximum \leq (19.8\ kg\ cm)$$

Two other constraints that can be considered are time and power. The designed robotic solutions should minimize both factors to produce efficient robots.

$$Time: Minimize$$

$$Power: Minimize$$

3.4 Evolutionary Algorithm (EA)

In order to find an optimal solution, combinatorial optimization techniques are used because the set of design parameters and search space are discrete in nature. Exhaustive search methods are not feasible in this case, because the search space is very large. Therefore, probabilistic search techniques, including Evolutionary Algorithms (EA), are applied to solve this problem. Evolutionary search techniques; such as Genetic Algorithm (GA) and Simulated Annealing (SA) methods are more proper to solve this kind of problems due to their capability of exploring large areas of the search space to avoid small local optima [52]. However, GA produced better results than SA when

applied to design problems generally and modular robotic design specifically, thus GA is used hereafter to solve the research problem [75-77].

GA is a probabilistic search algorithm and optimization method that simulates biological evolution in terms of selection, mutation, and crossover. GA is based on the concept of genetic reproduction to simulate evolution by selecting the fittest of population to be the parents and recombining them by performing crossover and then mutating the resulting children to produce the next generation [72, 78]. Figure 3.12 demonstrates the basic steps of GA.

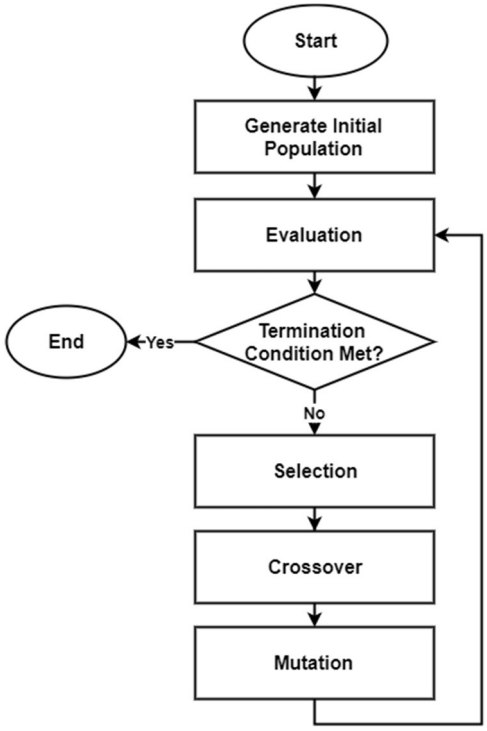


Figure 3.12. GA Flowchart

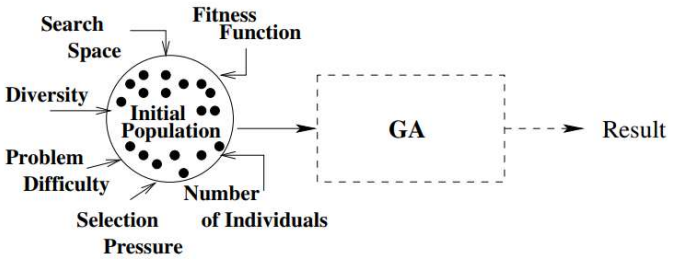


Figure 3.13. Some factors to take into account when the initial population is generated randomly. Image Courtesy of [82]

3.4.1 Generating Initial Population

The first step of a functioning GA is to create an initial population composed of individuals representing possible solutions to the problem. In this paper, AIM is used to represent each individual of every population. The AIM generating function involves generating the labels of connection sockets and the associated rotation angles.

The input to this function is an inventory of available link modules L and the maximum number of allowed link modules to build the robot n_max. The output is an AIM satisfying all constraints [72, 79-81].

When generating an initial population, a number of factors should be considered to ensure creating good individuals and finding an optimal solution accordingly. Among these factors are the fitness function, diversity, and number of individuals as shown in Figure 3.13 [82].

3.4.2 Evaluation

GA entails a fitness function to evaluate each individual in the current population. The fitness of an individual depends on how well it solves the problem at hand [83-84].

3.4.3 Selection

Selection is a mechanism for selecting individuals in the population for reproduction based on their fitness. Higher fitness values increase the likelihood of selecting an individual to reproduce and contribute offspring to the next generation [83-84].

3.4.4 Crossover

Crossover is the process of mimicking biological recombination between two individuals by combining their genetic information in order to create new offspring. Selecting the parents properly improves the resulting offspring [71, 83-84].

The crossover operator chooses a locus randomly and swaps the rows in the two parents AIMs below that locus to form two offspring; as demonstrated below. This operation is controlled by a crossover probability parameter $p_c \mid 0 \leq p_c \leq 1$. Values of p_c can include 0 and 1 because it is a good practice to allow some members of the current population survive to the next generation.

Before Swapping

$$A_1 = \begin{matrix} & e_1 \\ v_1 \\ v_2 \end{matrix}\begin{bmatrix} (1,0,0,0) \\ (4,90,0,0) \end{bmatrix} \qquad A_2 = \begin{matrix} & e_1 \\ v_1 \\ v_2 \end{matrix}\begin{bmatrix} (3,0,0,90) \\ (5,0,180,0) \end{bmatrix}$$

After Swapping

$$\acute{A}_1 = \begin{matrix} & e_1 \\ v_1 \\ v_2 \end{matrix}\begin{bmatrix} (1,0,0,0) \\ (5,0,180,0) \end{bmatrix} \qquad \acute{A}_2 = \begin{matrix} & e_1 \\ v_1 \\ v_2 \end{matrix}\begin{bmatrix} (3,0,0,90) \\ (4,90,0,0) \end{bmatrix}$$

3.4.5 Mutation

Mutation is the random deformation of individuals with a very small probability to ensure diversity and avoid falling into local maxima [68, 83-84].

The mutation operator makes small changes on non-zero entries of a single AIM including labels of connection sockets and symbols of links, but not types of links. This operation is controlled by a mutation probability parameter $p_m \mid 0 \leq p_m \leq 1$ and subject to all assembly rules.

3.5 Conclusion

This chapter presented several basic mathematical concepts for modular robotic modeling. Kinematic model and AIMs were discussed to describe the modular robotic assembly configuration. The modular robotic task-based design problem was formulated as a combinatorial optimization problem. Then, evolutionary algorithm was applied to

solve this problem. The next chapter describes how these elements come together to form the framework of the autonomous evolutionary modular robotic designer.

CHAPTER 4: SYSTEM IMPLEMENTATION

In the previous chapter, the mathematical model for the evolutionary task-based modular robotic design system was introduced. This chapter describes how these aspects are implemented in simulation to produce a feasible system that meets the dissertation objectives.

First, this chapter describes the genotype to phenotype mapping strategy used to evolve and produce optimal results. Then, it introduces the simulation framework that was used to simulate the modular robots and evaluate the evolved robotic structures by measuring their fitness values. Thereafter, the hardware implementation details of the physical modular robots are discussed. Finally, the test plan of the system is presented.

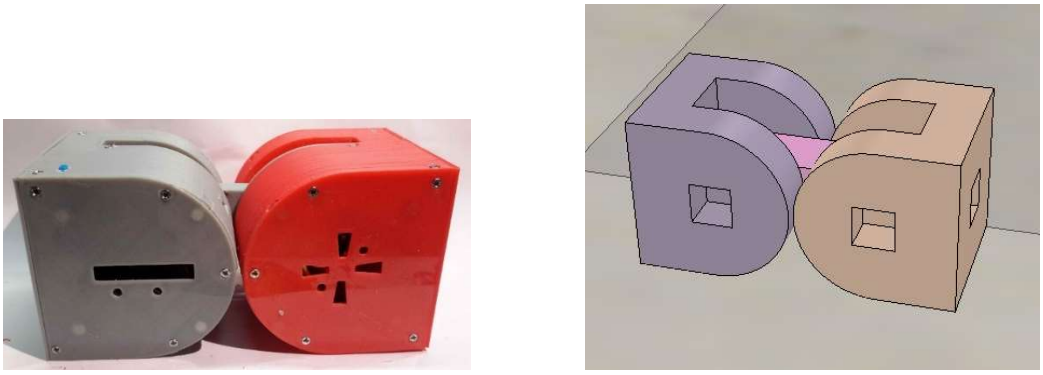
4.1 Genotype to Phenotype Mapping

Direct encoding is applied for genotype to phenotype mapping of the modular robots, because its capability of increasing the efficiency of artificial evolution compared to indirect encoding. It evolves every parameter of all simulated modules to construct a modular robot, which increases the search space but leads to better evolved robotic designs [85].

4.2 Simulation

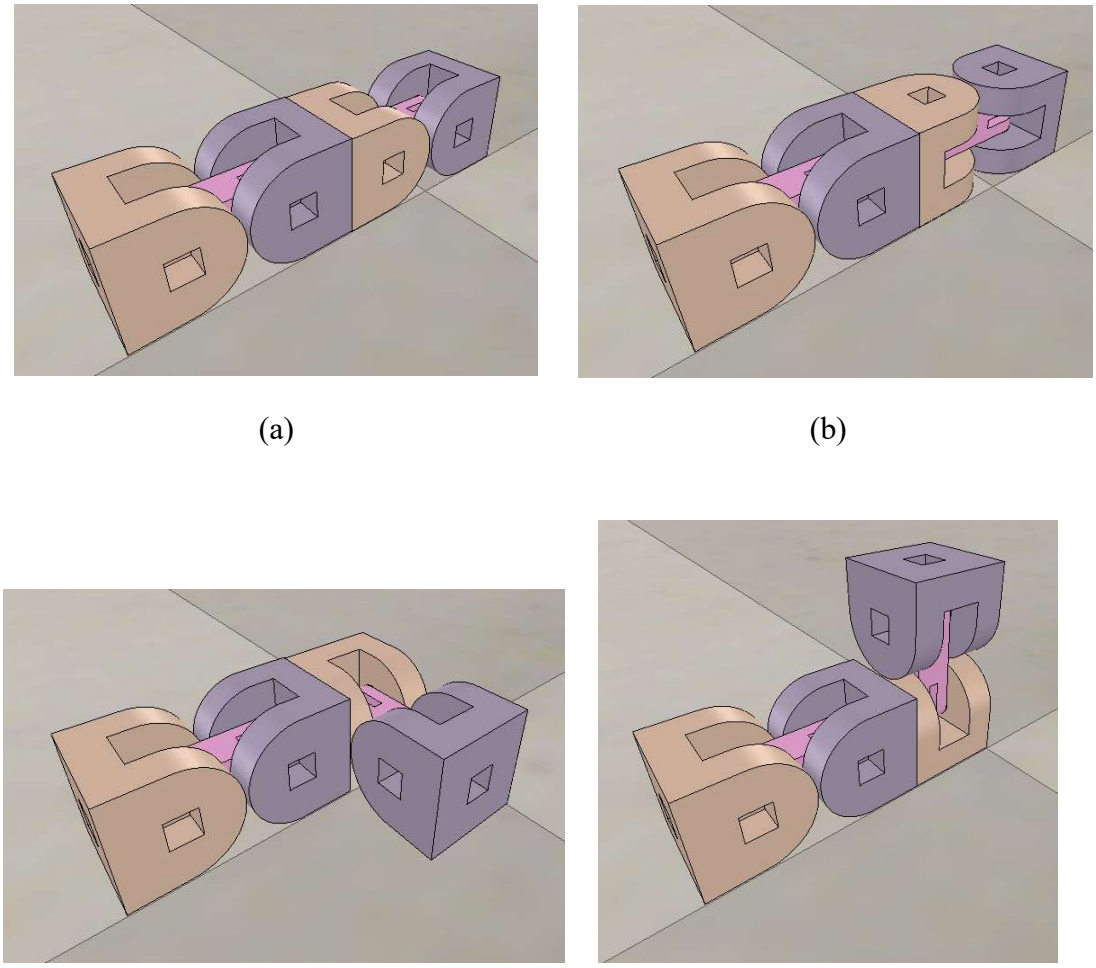
The Virtual Robot Experimentation Platform (V-REP) is a powerful, versatile, and scalable simulation framework that can efficiently perform algorithm optimization and modular robot simulation [86].

Dtto modules are modeled in V-REP according to their physical properties as illustrated in Figure 4.1. A physical module has magnets to connect to other modules. The connection breaking force and torque parameters are modeled in the simulated module. Moreover, every single link module consists of a male block, with three connection sockets, and a female block, with three connection sockets as well, to achieve a verity of connections as demonstrated in Figure 4.2.



(a) (b)

Figure 4.1. (a) Dtto module in real world (b) Dtto module in simulation



(a) (b)

(c) (d)

Figure 4.2. Variety of two module connections

4.2.1 Physics Engine

Bullet physics engine is an open source library that is supported by V-REP. It is used to simulate collision detection and rigid body dynamics. The reasons for choosing Bullet over other physics engines include its solid performance, high simulation fidelity, and scalability. Bullet provides good simulation speed on a single computer. It can achieve

a high degree of similarity between simulated modules and real modules and can simulate complex modules and environments. Finally, it can simulate a large number of modules simultaneously [87].

4.3 Hardware Implementation

Each Dtto module is comprised of 3D printed parts and electronic components as shown in Figure 4.3. The major components include five actuators: two MG92B servomotors and three SG90 servomotors. These actuators are controlled using Arduino Nano with ATmega328P CPU. For communication between multiple modules, the control circuit involves NRF24L01 wireless transceiver, and for communication with a host PC, HC-05 Bluetooth module is integrated in the circuit. Every module is powered by a Li-Po3.7v 600mAh battery.

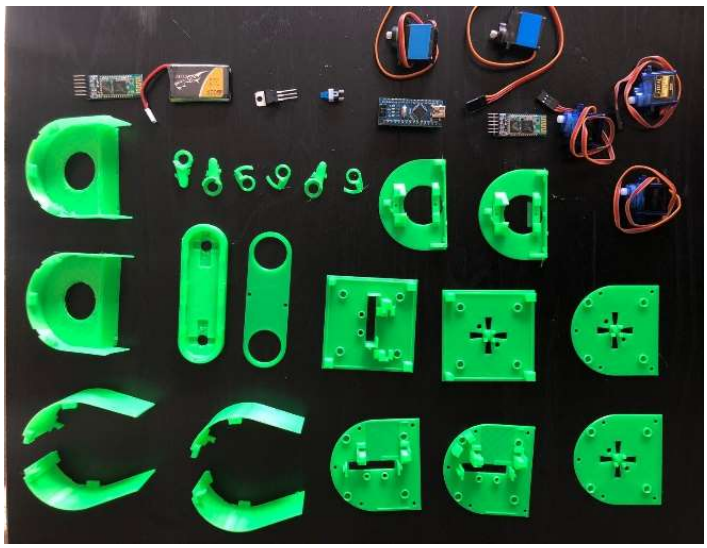


Figure 4.3. Components of a single Dtto module

4.4 System Testing

To validate the quality of the resulting robots, the linear mission benchmark robotic problem is considered. This problem involves several test scenarios ranging in difficulty from easy to hard in order to examine the different aspects of the proposed system. Thus, this system must produce robots that can accomplish the input tasks successfully.

4.4.1 Moving on a Flat Surface

The goal of the simulated robots is to move in a horizontal direction as far as possible from their initial position within 20 seconds of simulation time. The distance is measured by the distance that the initial module has traveled.

4.4.2 Obstacle Avoidance

The goal this time is to move and avoid obstacle to reach a destination within 50 seconds of simulation time. The distance is measured by the distance that the initial cube module has traveled.

4.5 Conclusion

This chapter described how the proposed system is implemented along with the used methods and technologies. It discussed the used simulator and its specific building blocks that were selected to produce the best possible results and maintain high performance. Finally, it discussed the test plan scenarios to be used in order to test the functionality of the implemented system.

CHAPTER 5: RESULTS

The results obtained using the proposed task-based evolutionary modular robotic design system focused on autonomously designing feasible homogeneous modular robots that can be built physically in real world using Dtto 3D printed modules.

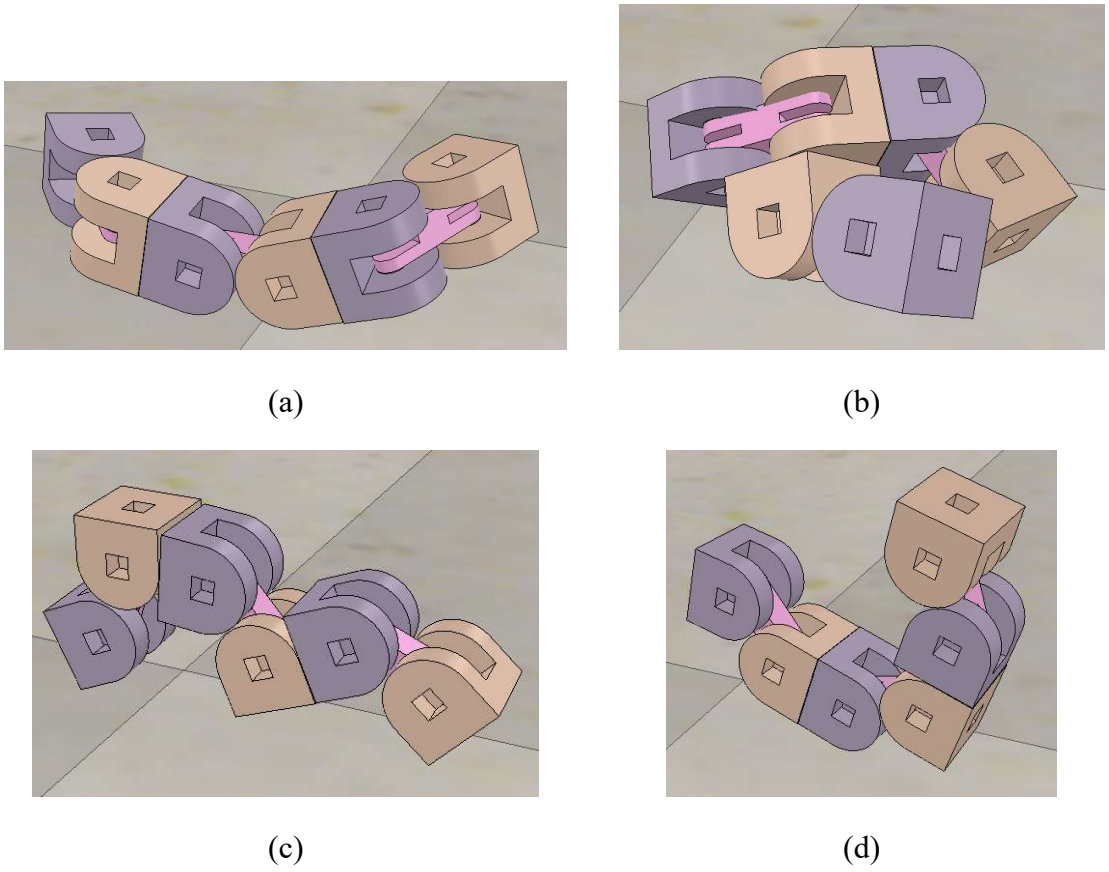
In this chapter, the results of running the test scenarios described in the previous chapter are discussed, followed by an analysis of configuration space enumeration and computational complexity.

The results of running the first test scenario described in the previous chapter, moving on a flat surface, were divided based on the maximum number of allowed link modules to build the robot n_{max}. Some results of autonomously designing modular robots using three and four Dtto modules are illustrated in Figure 5.1 and 5.2 respectively.

The obtained solutions were realistic and feasible. They can be implemented physically and built using Dtto modules in real life. However, some of these resulting robots are not practical as they did not follow the input trajectory successfully. Using a smaller number of modules resulted in better performing robots than using a larger number of modules. The problem comes from the lack of adequate constraints. Nevertheless, those solutions are appropriate because of their speed and scalability. The video demonstrating the results can be viewed here.

In order to run the second test scenario, moving on a flat surface and avoiding obstacles, a proximity sensor was added to the first module of the resulting robot. The robot

halts when it detects an obstacle and continues to move when the obstacle is removed, as illustrated in this video, that shows a snake robot composed of 4 Dtto module to test the obstacle detection function. The snake robot containing the proximity sensor is illustrated in Figure 5.3. In order to allow the robots to avoid obstacles, we replaced the wall with a smaller one while keeping the proximity sensor in the snake robot. The robot has successfully overcome that obstacle by moving over it and pursuing the original locomotion mode after avoiding the obstacles. The results are demonstrated in Figure 5.4.



(a) (b)

(c) (d)

Figure 5.1. Resulting evolved robots made of three Dtto modules

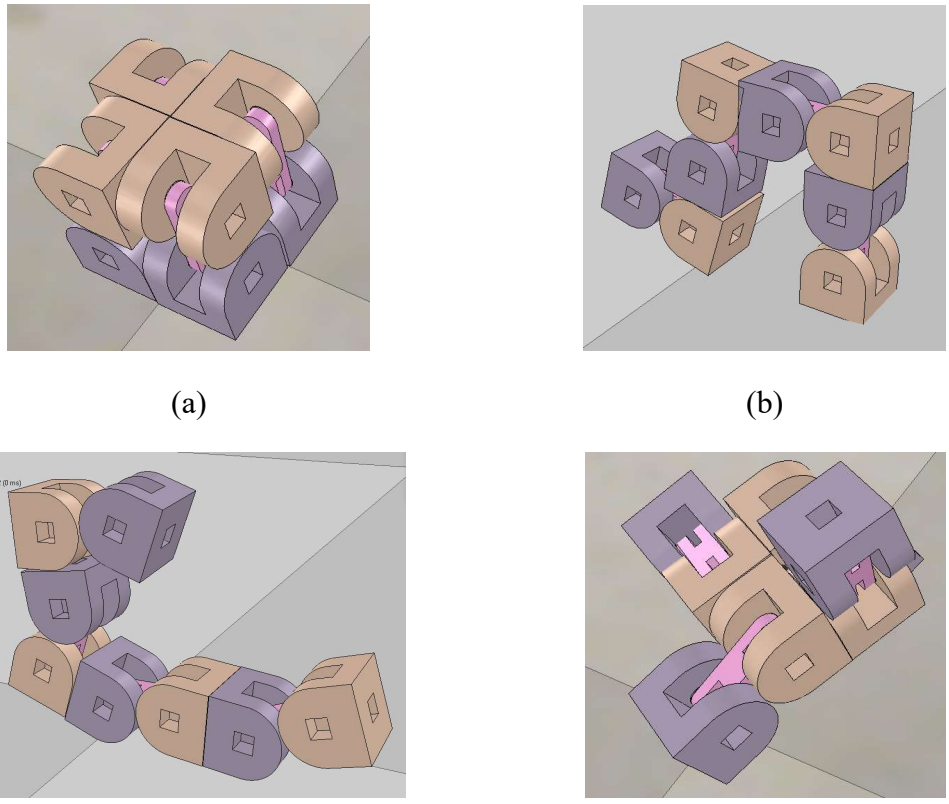


Figure 5.2. Resulting evolved robots made of Four Dtto modules

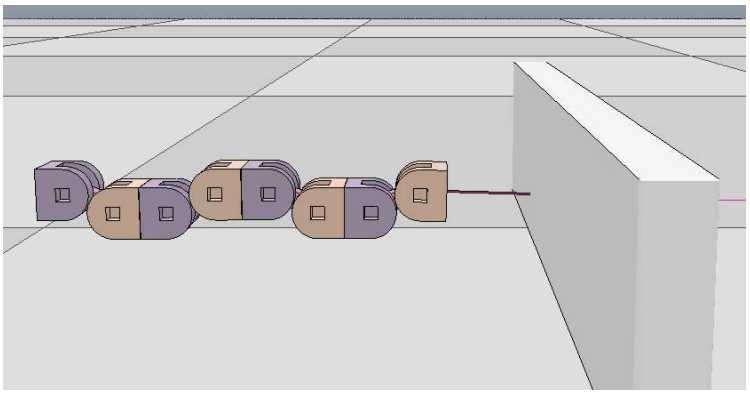
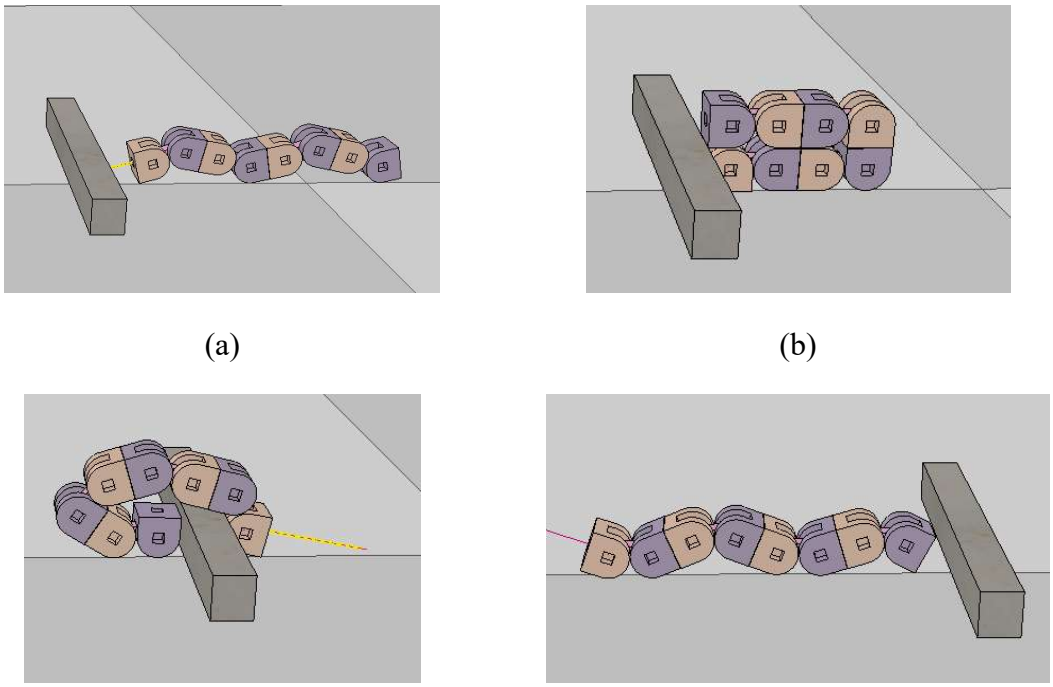


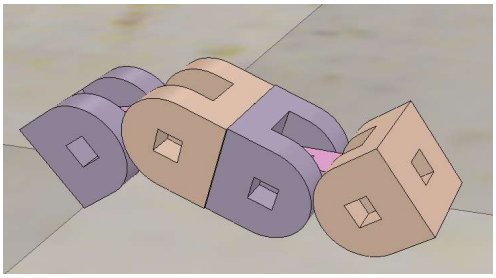
Figure 5.3. Snake robot with added proximity sensor



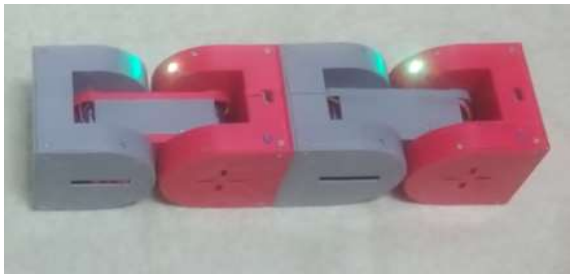
(a) (b)

(c) (d)

Figure 5.4. Snake robot crossing wall to demonstrate obstacle avoidance steps

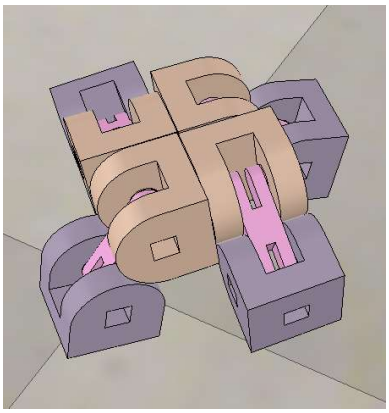


(a)



(b)

Figure 5.5. Dtto robot in simulation and reality – 2 modules



(a)

(b)

Figure 5.6. Dtto robot in simulation and reality – 4 modules

We chose the modular robots that had the best fitness results for physical implementation. The 1st one is composed of 2 Dtto modules, as shown in Figure 5.5. The 2nd one is composed of 4 modules, as illustrated in Figure 5.6. Both robots can be viewed in simulation and in reality, in this video[1].

[1] https://youtu.be/k2D0zBczNwo

5.1 Configuration Space Enumeration

Enumeration can be used to measure the configuration space in order to highlight the pros and cons of the different robotic designs. The calculated metric is non-isomorphic configurations that counts robotic designs with different modular orientation. The enumeration of the configuration space is shown in Table 5.1 and the five non-isomorphic configurations that can be obtained using two Dtto modules are demonstrated in Figure 5.7.

Table 5.1. Enumeration of the configuration space for Dtto modules

Number of modules (n)	***Number of non-isomorphic configurations (c)***	***Mean of computation time in ms (t)***
2	5	0.0
3	685	11.8
4	153,929	3437.1

Figure 5.7. All non-isomorphic configurations using 2 Dtto modules

Enumerating the configuration space of Dtto modules is computationally expensive and could not be extended beyond four modules because the workstation runs out of memory. We used an Intel® Core™ i5, 2.4 GHz CPU with 12 GB of RAM. All configurations were saved in memory to check for duplicate robotic configurations. The search space can be increased significantly by replacing the male and female connection faces with unisex ones.

5.2 Complexity Analysis

The complexity of designing modular robots grows exponentially with the number of modules used. This problem is computationally intractable, because the problem of enumerating all possible robotic configurations of n modules is related to the cell growth problem in graph theory and combinatorics, which is still an unsolved problem [88].

5.3 Conclusion

The results presented in this chapter covered 8 trials of the designer system and involved generating and simulating thousands of robotic configurations. The results verified the ability of the proposed system to generate feasible modular robotic designs. They also revealed some limitations that could be overcome in future work.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

Robots are essential to perform important tasks that are difficult and expensive to be undertaken by humans; such as space exploration, nuclear sites inspection, and hazardous rescue operations. However, these robotic applications are not epidemic because of their high cost and long development time. Therefore, modular robots are used to overcome the limitations of developing mainstream robots by offering a method of rapid and cost-effective production. Though, designing modular robotic systems to accomplish such sophisticated tasks in unstructured environments can be very challenging for human designers, especially with the lack of practical design guidelines. This difficulty increases the need for an autonomous system to design modular robots.

This dissertation proposed an autonomous task-based modular robotic design system based on an evolutionary approach to find the optimal results. The autonomous evolutionary designer takes as input the task to be performed by the resulting robot, and the modular repertoire. The input task is described as a motion trajectory describing the robot motion through space. The input modular data includes the number of available modules and the modular primitives, to allow extending the system to include a variety of modular geometries and DOFs. The system starts the evolutionary process by constructing the robotic individuals of the initial population. Each robot is represented as a graph that can be expressed as an Assembly Incidence Matrix (AIM). These robotic representations allow the system to be more flexible and include different types and shapes of modules. Then, the robotic individuals evolve in simulation to produce better solutions. The final

output of the system is a simulated robotic structure that is capable of performing the input task. This robot can be implemented in reality using physical hardware. In this research, Dtto modular robot is used to build the resulting robots.

The results were promising. Thus, we need to continue working on the same framework by adding more test scenarios and adding more modules to compare the results and the performance of the system. Then, we can proceed to the final step which is building the physical robots using Dtto 3D printed modules. Then, transferring the control from the simulated robot to the physical robot and compare the behavior of the robots. Moreover, we can add more modular types to test the system extensibility.

REFERENCES

[1] A. Faíña, D. Souto, F. Orjales, F. Bellas, and R.J. Duro, "From Virtual Creatures to Feasible Robots," In Genetic and Evolutionary Computation Conference GECCO'14, pp. 340-354, 2014.

[2] J. C. Bongard, "Evolutionary robotics," Communications of the ACM, vol. 56, no. 8, pp. 74-83, 2013.

[3] H. Lipson, "Uncontrolled engineering: A review of S. Nolfi and D. Floreano's evolutionary robotics," Artificial Life, vol. 4, no. 7, pp. 419-424, 2000.

[4] J.C. Bongard, "The 'what', 'how' and the 'why' of evolutionary robotics," New Horizons in Evolutionary Robotics, Springer, pp. 29-35, 2011.

[5] F. Silva, M. Duarte, L. Correia, S.M. Oliveira, and A.L. Christensen, "Open issues in evolutionary robotics," Evolutionary computation, vol. 24, no. 2, pp.205-236, 2016.

[6] R. Alattas, "Analyzing Modular Robotic Systems," in Online Engineering & Internet of Things, Springer, pp. 1014-1028, 2018.

[7] A. Faíña, F. Bellas, F. López-Peña, and R.J. Duro, "EDHMoR: Evolutionary designer of heterogeneous modular robots," in Engineering Applications of Artificial Intelligence, vol. 26 no. 10, pp. 2408-2423, 2013.

[8] D. Floreano, P. Husbands, and S. Nolfi, "Evolutionary Robotics," Handbook of Robotics, Chapter 61, 2007.

[9] D.A. Sofge, M.A. Potter, M.D. Bugajska, and A.C. Schultz, "Challenges and opportunities of evolutionary robotics," In Proceedings of the Second International

Conference on Computational Intelligence, Robotics and Autonomous Systems CIRAS'03, 2003.

[10] S. Nolfi and D. Floreano, "Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines," Bradford Books, 2004.

[11] H. Lipson, "Evolutionary robotics and open-ended design automation," Biomimetics, vol. 17, no. 9, pp. 129-155, 2005.

[12] C. Paul and J.C. Bongard, "The Road Less Traveled: Morphology in the Optimization of Biped Robot Locomotion", in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems IROS'01 Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180), vol. 1, pp. 226-232, 2001.

[13] V. Zykov, J.C. Bongard., and H. Lipson, "Evolving Dynamic Gaits on a Physical Robot", in Proceedings of Genetic and Evolutionary Computation Conference GECCO'04, 2004.

[14] H.H. Lund, J. Hallam and W.P. Lee, "Evolving robot morphology," IEEE International Conference on Evolutionary Computation, pp. 197-202, 1997.

[15] H.H. Lund, "Co-evolving control and morphology with Lego robots," In Morpho-functional Machines: The New Species, pp. 59-79, 2003.

[16] K. Sims, "Evolving 3D morphology and behavior by competition," in Artificial Life, vol. 1, no. 4, pp. 28–39, 1994.

[17] H. Lipson and J.B. Pollack, "Automatic design and manufacture of robotic lifeforms," in Nature, vol. 406, no. 6799, pp. 974-978, 2000.

[18] M. Yim, D. Duff, and K. Roufas, “PolyBot: A Modular Reconfigurable Robot,” in IEEE International Conference on Robotics and Automation ICRA’00, pp. 514–520, 2000.

[19] I.M. Chen and J.W. Burdick, “Determining Task Optimal Modular Robot Assembly Configurations,” in IEEE International Conference on Robotics and Automation ICRA’95, vol. 1, pp. 132-137, 1995.

[20] W.K. Chung, J. Han, Y. Youm, and S.H. Kim, “Task based design of modular robot manipulator using efficient genetic algorithm,” in IEEE International Conference on Robotics and Automation ICRA’97, vol. 1, pp. 507-512, 1997.

[21] O. Chocron and P. Bidaud, “Evolutionary algorithms in kinematic design of robotic systems,” in Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS’97, vol. 2, pp. 1111-1117, 1997.

[22] G. Yang and I.M. Chen, “Task-based optimization of modular robot configurations: minimized degree-of-freedom approach,” Mechanism and machine theory, vol. 35, issue. 4, pp.517-540, 2000.

[23] G.S. Hornby, H. Lipson, and J.B. Pollack, “Evolution of generative design systems for modular physical robots,” in Proceedings of IEEE International Conference on Robotics and Automation ICRA’01, vol. 4, pp. 4146-4151, 2001.

[24] J. Bishop, S. Burden, E. Klavins, R. Kreisberg, W. Malone, N. Napp, and T. Nguyen, “Self-organizing programmable parts,” in IEEE/RSJ International Conference on Intelligent Robots and Systems IROS’05, pp. 3684-3691, 2005.

[25] E. Bonabeau, M. Dorigo, and G. Theraulaz, “Swarm Intelligence: From Natural to Artificial Systems,” Oxford University Press, 1999

[26] P.J. White, K. Kopanski, and H. Lipson, “Stochastic self-reconfigurable cellular robotics,” in Proceedings of IEEE International Conference on Robotics and Automation ICRA’04, vol. 3, pp. 2888-2893, 2004.

[27] M. Tolley, J.D. Hiller, and H. Lipson, “Evolutionary design and assembly planning for stochastic modular robots,” in New Horizons in Evolutionary Robotics, Springer, pp. 211–225, 2011.

[28] P. White, V. Zykov, J.C. Bongard, and H. Lipson, “Three Dimensional Stochastic Reconfiguration of Modular Robots,” in Robotics: Science and Systems, pp. 161-168, 2005.

[29] M. Yim, Y. Zhang, and D. Duff, “Modular robots,” IEEE Spectrum, vol. 39, no. 2, pp. 30-34, 2002.

[30] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins and G. Chirikjian, “Modular Self-Reconfigurable Robot Systems [Grand Challenges of Robotics],” IEEE Robotics & Automation Magazine, vol. 14, no. 1, pp. 43-52, 2007.

[31] E.H. Østergaard, K. Kassow, R. Beck, and H.H. Lund, “Design of the ATRON lattice-based self-reconfigurable robot,” in Autonomous Robots, vol. 21, no. 2, pp. 165–183, 2006.

[32] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, “M-TRAN : Self-Reconfigurable Modular,” in IEEE/ASME transactions on Mechatronics, vol. 7, no. 4, pp. 431–441, 2002.

[33] H. Kurokawa, A. Kamimura, E. Yoshida, K. Tomita, S. Kokaji, and S. Murata, “M-TRAN II: Metamorphosis from a four-legged walker to a caterpillar,” in Intelligent Robots and Systems IROS’03, vol. 3, pp. 2454-2459, 2003.

[34] M. Yim, C. Eldershaw, Y. Zhang, and D. Duff, “Self-reconfigurable robot systems: PolyBot,” Journal of the Robotics Society of Japan, vol. 21, no. 8, pp. 851-854, 2003.

[35] A. Golovinsky, M. Yim, Y. Zhang, C. Eldershaw, and D. Duff, “PolyBot and PolyKinetic™ System: a modular robotic platform for education,” in Proceedings of IEEE International Conference on Robotics and Automation ICRA’04, vol. 2, pp. 1381-1386, 2004.

[36] D. Duff, M. Yim, and K. Roufas, “Evolution of PolyBot: A modular reconfigurable robot,” in Proceedings of the Harmonic Drive International Symposium, 2001.

[37] M. Yim, Y. Zhang, K. Roufas, D. Duff, and C. Eldershaw, “Connecting and disconnecting for chain self-reconfiguration with PolyBot,” in IEEE/ASME Transactions on Mechatronics, vol. 7, no. 4, pp. 442–451, 2002.

[38] J. Zhao, X. Cui, Y. Zhu, and S. Tang, “A new self-reconfigurable modular robotic system UBot: Multi-mode locomotion and self-reconfiguration,” in IEEE International Conference on Robotics and automation ICRA’11, pp. 1020-1025, 2011.

[39] T. Fukuda and Y. Kawauchi, “Cellular robotic system (CEBOT) as one of the realizations of self-organizing intelligent universal manipulator,” in IEEE International Conference on Robotics and Automation ICRA’90, pp. 662–667, 1990.

[40] T. FUKUDA and S. NAKAGAWA, “Method of autonomous approach, docking and detaching between cells for dynamically reconfigurable robotic system CEBOT,” JSME international journal Ser. 3, Vibration, control engineering, engineering for industry, vol. 33, no. 2, pp. 263-268, 1990.

[41] T. Fukuda, M. Buss, H. Hosokai, and Y. Kawauchi, “Cell Structured robotic system CEBOT: Control, planning and communication methods,” Robotics and autonomous systems, vol. 7, no. 2–3, pp. 239–248, 1991.

[42] E.H. Østergaard, and H.H. Lund, “Evolving control for modular robotic units,” in Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA’03, pp. 886–892, 2003.

[43] S. Haghzad, S. Bagheri, and S. Faraji, “Finding proper configurations for modular robots by using genetic algorithm on different terrains,” International Journal of Materials, Mechanics and Manufacturing, pp.360-365, 2013.

[44] V. Zykov, E. Mytilinaios, M. Desnoyer, and H. Lipson, “Evolved and designed self-reproducing modular robotics,” in IEEE Transactions on Robotics, vol. 23, no. 2, pp. 308-319, 2007.

[45] D. Rus and M. Vona, “Crystalline robots: Self-reconfiguration with compressible units modules,” in Autonomous Robots (special issue on Modular Reconfigurable Robots), vol. 10, no. 1, pp. 107–124, 2001.

[46] W. Suh, S.B. Homans, and M. Yim, “Telecubes: mechanical design of a module for self-reconfigurable robotics,” in Proceedings of IEEE International Conference on Robotics and Automation ICRA’02, vol. 4, no. 5, pp. 4095–4101, 2002.

[47] S. Vassilvitskii, M. Yim, and J. Suh, “A complete, local and parallel reconfiguration algorithm for cube style modular robots,” in Proceedings of IEEE International Conference on Robotics and Automation ICRA’02, vol. 1, no. 5, pp. 117–122, 2002.

[48] H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata, "Distributed self-reconfiguration of M-TRAN III modular robotic system," in The International Journal of Robotics Research, vol. 2, no. 3-4, pp. 373-386, 2008.

[49] M. Yim, "A reconfigurable modular robot with many modes of locomotion," in Proceedings of the JSME International Conference on Advanced Mechatronics, pp. 283–288, 1993.

[50] J. Bishop, S. Burden, E. Klavins, R. Kreisberg, W. Malone, N. Napp, and T. Nguyen, "Programmable parts: A demonstration of the grammatical approach to self-organization," in IEEE/RSJ International Conference on Intelligent Robots and Systems IROS'05, pp. 2644–2651, 2005.

[51] N. Napp, S. Burden, and E. Klavins, "The statistical dynamics of programmed self-assembly," in Proceedings of IEEE International Conference on Robotics and Automation ICRA'06, pp. 1469–1476, 2006.

[52] V. Zykov, A. Chan, and H. Lipson, "Molecubes: An open-source modular robotics kit," in International Conference on Intelligent Robots and Systems IROS'07 Self-Reconfigurable Robotics Workshop, pp. 3-6, 2007.

[53] V. Zykov, W. Phelps, N. Lassabe, and H. Lipson, "Molecubes extended: Diversifying capabilities of open-source modular robotics," in International Conference on Intelligent Robots and Systems IROS'08 Self-Reconfigurable Robotics Workshop, pp. 22-26, 2008.

[54] H. Zhong, Z. Li, H. Zhang, C. Yu, and N. Li, "Modular robot path planning using genetic algorithm based on gene pool," Advances in Computation and Intelligence, pp.380-389, 2010.

[55] G.G. Ryland, and H.H. Cheng, “Design of iMobot, an intelligent reconfigurable mobile robot with novel locomotion,” in IEEE International Conference on Robotics and Automation ICRA’10, pp. 60-65, 2010.

[56] D. Ko, and H.H. Cheng, “Reconfigurable software for reconfigurable modular robots,” in Proceedings of International Conference on Robotics and Automation ICRA’10 Workshop Modular Robots: State of the Art, p. 100, 2010.

[57] J. Zhao, X. Cui, Y. Zhu, and S. Tang, “UBot: a new reconfigurable modular robotic system with multimode locomotion ability,” Industrial Robot: An International Journal,”, vol. 39, no. 2, pp.178-190, 2012.

[58] Y. Zhu, J. Zhao, X. Cui, X. Wang, S. Tang, X. Zhang, and J. Yin, “Design and implementation of ubot: A modular self-reconfigurable robot,” in IEEE International Conference on Mechatronics and Automation ICMA’13, pp. 1217-1222, 2013.

[59] J. Davey, N. Kwok, and M. Yim, “October. Emulating self-reconfigurable robots-design of the SMORES system,” in IEEE/RSJ International Conference on Intelligent Robots and Systems IROS’12, pp. 4464-4469, 2012.

[60] T. Tosun, G. Jing, H. Kress-Gazit, and M. Yim, “Computer-aided compositional design and verification for modular robots,” In Robotics Research, Springer, pp. 237-252, 2018.

[61] J.B. Pollack, H. Lipson, S. Ficici, P. Funes, G. Hornby, and R.A. Watson, “Evolutionary techniques in physical robotics,” in International Conference on Evolvable Systems, pp. 175-186, 2000.

[62] J. Auerbach, D. Aydin, A. Maesani, P. Kornatowski, T. Cieslewski, G. Heitz, P. Fernando, I. Loshchilov, L. Daler, and D. Floreano, “Robogen: Robot generation

through artificial evolution," in Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems, the MIT Press, pp. 136-137, 2014.

[63] E. Samuelsen, and K. Glette, "Real-world reproduction of evolved robot morphologies: automated categorization and evaluation," in European Conference on the Applications of Evolutionary Computation, Springer, pp. 771-782, 2015.

[64] D. Cellucci, R. MacCurdy, H. Lipson, and S. Risi, "1D Printing of Recyclable Robots," IEEE Robotics and Automation Letters, vol. 2, no. 4, pp.1964-1971, 2017.

[65] S. Murata, H. Kurokawa, and S. Kokaji, "Self-assembling machine," in Proceedings of the IEEE International Conference on Robotics and Automation ICRA'94, pp. 441-448, 1994.

[66] G. S. Chirikjian, "Metamorphic Hyper-Redundant Manipulators," in Proceedings of JSME International Conference on Advanced Mechatronics, pp. 467-472, 1993.

[67] J. W. Suh, S. B. Homans, and M. Yim, "Telecubes: Mechanical Design of a Module for Self-Rreconfigurable Robotics," in Proceedings of IEEE International Conference on Robotics and Automation ICRA'02, vol. 4, no. 5, pp. 4095–4101, 2002.

[68] N. Correll, C. Wailes, and S. Slaby. "A One-hour Curriculum to Engage Middle School Students in Robotics and Computer Science using Cubelets" in Distributed Autonomous Robotic Systems, pp. 165-176. Springer, 2014.

[69] Dtto - Exploere Modular Robot: https://hackaday.io/project/9976-dtto-explorer-modular-robot [Accessed Jan 2018].

[70] J.W. Romanishin, K. Gilpin, and Daniela Rus, “M-blocks: Momentum-driven, magnetic modular robots,” in IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4288-4295, 2013.

[71] I.M. Chen, and J.W. Burdick, “Enumerating the non-isomorphic assembly configurations of modular robotic systems,” The International Journal of Robotics Research, vol. 17, no. 7, pp.702-719, 1998.

[72] G. Yang, and I.M. Chen, “Task-based optimization of modular robot configurations: minimized degree-of-freedom approach,” Mechanism and machine theory, vol. 35, no. 4, pp.517-540, 2000.

[73] A.L. Nelson, G.J. Barlow, and L. Doitsidis, “Fitness functions in evolutionary robotics: A survey and analysis,” Robotics and Autonomous Systems, vol. 57, no. 4, pp. 345-370, 2009.

[74] I. Chen, “Theory and applications of modular reconfigurable robotic systems,” Doctoral dissertation, California Institute of Technology, 1994.

[75] T.W. Manikas, and J.T. Cain, “Genetic algorithms vs. simulated annealing: A comparison of approaches for solving the circuit partitioning problem,” Technical report, University of Pittsburgh 1996.

[76] D. Marbach and A.J. Ijspeert, “Online optimization of modular robot locomotion,” In IEEE International Conference on Mechatronics and Automation ICMA’05, vol. 1, pp. 248-253, 2005.

[77] J.O. Kim and P. Khosla, “Design of Space Shuttle Tile Servicing Robot: An Application of Task Based Kinematic Design,” Proceedings of IEEE International Conference on Robotics and Automation ICRA’93, pp 867-874, 1993.

[78] P. Ross, and D. Corne, “Applications of genetic algorithms. AISB Quarterly on Evolutionary Computation,”, vol. 89, pp.23-30, 1994.

[79] J. Rieffel, “Evolutionary fabrication: the co-evolution of form and formation,” Doctoral Dissertation, Brandeis University, 2006.

[80] Y.C. Lin, K.S. Hwang, and F.S. Wang, “A mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems,” Computers & Mathematics with Applications, vol. 47 no. 8-9, pp.1295-1307, 2004.

[81] T. Ueyama, T. Fukuda and F. Arai, “Structure Configuration Using Genetic Algorithm for Cellular Robotic System,” Proceedings of International Conference on Intelligent Robots and Systems IROS’92, Raleigh, NC, pp 1542-1549, 1992.

[82] P.A. Diaz-Gomez, and D.F. Hougen, “Initial Population for Genetic Algorithms: A Metric Approach,” In Proceedings of the International Conference on Genetic and Evolutionary Methods GEM’07, pp. 43-49, 2007.

[83] S.M. Thede, “An introduction to genetic algorithms,” Journal of Computing Sciences in Colleges, vol. 20, no. 1, pp.115-123, 2004.

[84] H. Yang, J. Mathew, and L. Ma, “Intelligent diagnosis of rotating machinery faults-a review,” In 3rd Asia-Pacific Conference on Systems Integrity and Maintenance, ACSIM’02, pp. 385-392, 2002.

[85] F. Veenstra, A. Faina, S. Risi, and K. Stoy, "Evolution and morphogenesis of simulated modular robots: a comparison between a direct and generative encoding," In European Conference on the Applications of Evolutionary Computation, pp. 870-885, Springer, 2017.

[86] E. Rohmer, S.P. Singh, S.P. and M. Freese, “V-REP: A versatile and scalable robot simulation framework,” In IEEE/RSJ International Conference on Intelligent Robots and Systems IROS’13, pp. 1321-1326, 2013.

[87] T. Collins, N.O. Ranasinghe, and W.M. Shen, “ReMod3D: A high-performance simulator for autonomous, self-reconfigurable robots,” In IEEE/RSJ International Conference on Intelligent Robots and Systems IROS’13, pp. 4281-4287, 2013.

[88] E.V. Konstantinova, “A survey of the cell-growth problem and some its variations,” Com 2 MaC-KOSEF, pp. 01-06, 2001.